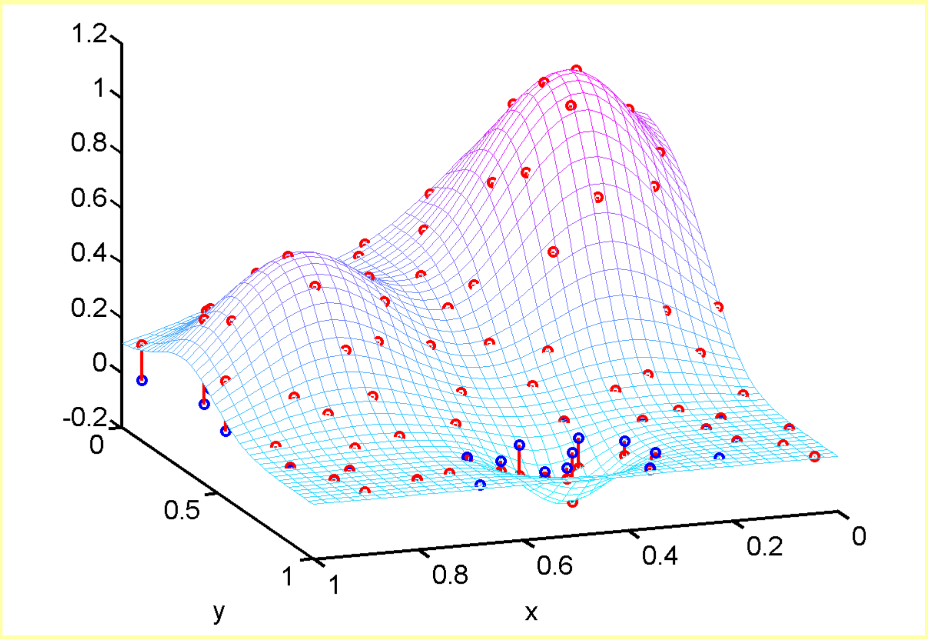
Lectures on Radial Basis Functions

by **Stefano De Marchi and Emma Perracchione**
Department of Mathematics "Tullio Levi-Civita"
University of Padua (Italy)



January 24, 2018

These lecture notes were inspired mainly by two seminal books on the topic by Holger Wendland [74] and by Gregory E. Fasshauer [32]. The first presents more theoretical aspects while the second provides also useful Matlab functions for understanding better the theory and all the applictions discussed. The notes have then been used during a short teaching-visit of the author to the University of Antwerp, for the Erasmus Teaching Staff Mobility.

People interested on radial basis functions, can refer to the wide literature available that, especially in the last two decades, has grown very fast. The popularity of radial basis functions can be understood by means of the following ”parallelism”. In many cooking recepies the parsley is used to give flavour and colour to dishes. Radial basis functions can be considered as a *mathematical parsley* since they have been used in all mathematical problems requiring a powerful, i.e. efficient and stable, approximation tool.

These lectures were thoughts for students without a strong background on functional analysis, so in the presentation of the topics we deliberately avoid, when possible, to introduce functional analysis concepts. This is a great lack, but we hope that the people who will use these notes will be not too critical to us.

Moreover, these are only **introductory lectures** and some **examples** on the topic. Many important aspects and applications, for lack of time and because there are many other books (see e.g. [74, 32, **?**]), are not considered. Every lecture provides also a set of exercises solvable by using Matlab. This choice has been done with the aim of making the discussion more interesting from both the numerical and geometrical point of view.

We do hope that after this brief introduction, interested students will be encouraged and also interested in getting into this fascinating mathematical tool that has become more and more popular in used and popular in many fields.

Stefano De Marchi and Emma Perracchione
Padova, February 2018.

Contents

List of Figures

List of Tables

Introduction

These lectures focus on recent theoretical and computational aspects (implemented with the Matlab software) in the field of *scattered data approximation.* More precisely, our attention is devoted to *meshfree* or *meshless* methods. Indeed, taking advantage of being independent of a mesh, they turn out to be truly performing in high dimensions. Unfortunately, not all of them allow to deal with a *large* number of points. Therefore, we also review the basic theory concerning local methods and in particular the Partition of Unity (PU) method. Furthermore, we give a general outline of collocation methods via RBF.

Historical remarks

Even if here we consider meshfree methods in the context of approximation theory, we have to point out that they have been used for the first time in statistics. In fact, their study began in 1870 with the work of W.S.B. Woolhouse [76]. It has been further developed few years later by E.L. De Forest [21, 22] and in 1883 by J.P. Gram [38]. Such works gave rise to *moving least squares* approximation (also known as *local regression* in statistic literature).

For what concerns approximation theory, the historical and theoretical foundation of meshless methods lies in the concept of *positive definite functions* or, more in general, *positive definite kernels.* Their development can be traced back to both the work of J. Mercer (1909) [57], a Fellow of Trinity College at Cambridge University, and the one of M. Mathias (1923) [50], who was a student at the University of Berlin. Their studies have been linked to Fourier transforms in the 1930s by S. Bochner [5], lecturer at the University of Munich at that time, and by I. Schoenberg [66], who was assistant professor at Colby College. Further studies about positive definite functions up to the mid 1970s have been collected by J. Stewart [71]. Later, C.A. Micchelli in 1986 made the connection between scattered data interpolation and positive definite functions [59]. At the same time, E.J. Kansa gave rise to the first methods for solving PDEs via meshless methods [48].

Many positive definite functions are nowadays classified as RBFs. Such term appeared for the first time in a publication made by N. Dyn and D. Levin in 1983 [30]. However, earlier in the 1970s, many researchers introduced functions that are now called RBFs. For instance, the term *multiquadrics* is due to R. Hardy, geodesist at Iowa State University [40].

Moreover, in 1972 R.L. Harder and R.N. Desmarais, two aerospace engineers, introduced the Thin Plate Splines (TPSs) [39]. Always at the same time, such functions appeared in several papers by J. Duchon, a mathematician at the Université Joseph Fourier in Grenoble [27, 28, 29]. Furthermore, we also have to mention J. Meingnet who, working at the Université Catholique de Luouvain, in 1970s introduced what are now known as polyharmonic splines [52, 53, 54, 55]. However, in the theory of RBFs the main step forward has been carried out by R. Franke in the 1980s. He proposed a comprehensive study about several methods for multivariate interpolation [34, 35].

Finally, concerning local techniques, we can affirm that the pioneer of the PU scheme has been D. Shepard. In fact, in the late 1960s he introduced, as an undergraduate student at Harvard University, what are now called the *Shepard weights* [69]. His studies have been motivated by the fields of geophysics and meteorology. Moreover, related to the same topics, we also have to mention few papers that contain sentences and traces which seem to indicate that the Shepard's scheme was known before his work (see the paper by I.K. Crain and B.K. Bhattacharyya (1967) [18] or the one by W.R. Goodin et al. [36]).

Aside from this short historical remark, in this dissertation we mainly refer to the recent books by M.D. Buhmann, G.E. Fasshauer, M.J. McCourt and H. Wendland [8, 32, 33, 74].

Lecture 1

Learning from splines

1.1 Motivations

In practical applications we have to face the problem of reconstructing an unknown function f from a set (usually small) of data. These data consist of two sets: the *data sites* $X = \{x_1, \ldots, x_N\}$ and the *data values* $f_j = f(x_j),\ j = 1, \ldots, N$. The reconstruction has to approximate the data values at the data sites. In practice we are looking for a function s that either *interpolates* the data, i.e. it satisfies the conditions $s(x_j) = f_j,\ 1 \le j \le N$ or *approximate* the data, i.e. $s(x_j) \approx f_j$. This latter is important, for example, when the data come from some measurement or contain noise.

In many cases the data are *scattered*, that is they have no special structure, and they are in a big amount (several millions). Moreover in several applications the data sites are considered in high dimension. Hence, for a unifying approach, methods have been developed in the last decades with the aim to meet all these (new) situations.

$\diamond\diamond$

We start from the univariate setting. We suppose that the data sites are ordered as follows

$$X : a < x_1 < x_2 < \cdots < x_N < b \tag{1.1}$$

and we have some data values $f_1, \ldots, f_N$ to be interpolated at the data set X. What we want to do, mathematically speaking, is finding $s : [a,b] \to \mathbb{R}$ with the property $s(x_j) = f_j$ for all $j = 1, \ldots, N$.

Notice, that the data values f_j is not necessary stem from a function f but we shall keep in mind this possibility for reasons that will become clearer later.

In the *univariate* setting, a simple solution of the above problem consists in taking s as polynomial p of degree at most $N-1$. However, as we can see later, this solution is not working in higher dimensions. Remaining in the univariate case, **no one with**

experience in approximation theory would even try to interpolate a hundred thousand points with a polynomial. Indeed it is a well-established fact that a large data set is better dealt with *splines* than by polynomials. One aspect to notice in contrast to polynomials, the accuracy of the interpolation process using splines is not based on the polynomial degree but on the spacing of the data sites.

Let us review briefly the main properties of univariate splines, especially in the case of *cubic splines.* The set of cubic splines corresponding to the subdivision (1.1) is the space

$$\mathcal{S}_3(X) = \{s \in \mathcal{C}^2[a,b] : s_{|[x_i,x_{i+1}]} \in \mathbb{P}_3(\mathbb{R}),\ 0 \le i \le N\} \tag{1.2}$$

with $a = x_0$ and $x_{N+1} = b$. The space $S_3(X)$ has dimension $N+4$, so that the interpolation conditions $s(x_i) = f_i,\ 1 \le i \le N$ are not sufficient to guarantee a unique interpolant. To enforce uniqueness, in the case of *natural splines*, i.e. the set

$$\mathcal{N}_3(X) = \{s \in \mathcal{S}_3(X) : s_{|[a,x_1]}, s_{|[x_N,b]} \in \mathbb{P}_1(\mathbb{R})\} \tag{1.3}$$

that consists of all cubic splines that are linear polynomials on the outer intervals $[a, x_1]$ and $[x_N, b]$. It come easy to see that a cubic spline s is a natural spline if and only if it satisfies $s''(x_1) = s^{(3)}(x_1) = 0$ and $s''(x_N) = s^{(3)}(x_N) = 0$. With this choice we have imposed 4 additional conditions to the space, so it is natural to assume that the $\texttt{dim}(\mathcal{N}_3(X)) = N$. Even more, it can be shown that the initial interpolation problem has a unique solution in $\mathcal{N}_3(X)$.

Here some important properties of splines that are worth to be mentioned

1. They are piecewise polynomials.
2. An interpolating natural cubic spline satisfies a *minimal norm property.* Assume that f comes from the *Sobolev space* $\mathcal{H}^2[a,b]$, i.e. f is continuous in $[a,b]$ and has weak first and second order derivatives in $L_2[a,b]$ (a more precise definition will be done later or can be found in any books of functional analysis). Assume further that f is such that $f(x_j) = f_j, 1 \le j \le N$. If $s_{f,X}$ denotes the natural cubic spline interpolant (at the data set X) then

   $$(f'' - s''_{f,X}, s''_{f,X})_{L_2[a,b]} = 0\,.$$

   This leads to the Pythagorean equation

   $$\|f'' - s''_{f,X}\|^2_{L_2[a,b]} + \|s''_{f,X}\|^2_{L_2[a,b]} = \|f''\|^2_{L_2[a,b]}\,,$$

   indicating that the natural cubic splines interpolant is the function from $\mathcal{H}^2[a,b]$ that minimizes the semi-norm $\|f''\|^2_{L_2[a,b]}$ under the conditions $f(x_j) = f_j$, $1 \le j \le N$.
3. They possess a local basis called *B-splines.* This basis, which is more stable than any other, can be defined by recursion, by divided differences of the *truncated cubic power* $p(x;t) = (x-t)^3_+$ or by convolution. Here x_+ takes the value of x for nonnegative x and zero otherwise.

Interested readers on splines and their many properties can refer to the following fundamental books by Schumaker [67] or de Boor [23].

Remarks

- Property 1. combined with the local basis, not only allows the efficient computation and evaluation of splines but also is the key ingredient for a simple error analysis. Hence, **the natural way of extending splines to the multivariate setting is based on this property**. To this end, a bounded region $\Omega \subset \mathbb{R}^d$ is partitioned into essentially disjoint regions $\{\Omega_j\}_{j=1}^N$ (patches). Then the spline space consists of those functions s that are piecewise polynomials on each Ω_j and that have smooth connections on the boundaries of two adjacent patches. In two dimensions the most popular subdivision of a polygonal region is by triangles. It is interesting to note that even in this simple case, the dimension of the spline space is in general unknown. When coming to higher dimensions it is not all clear what an appropriate replacement for the triangulation would be. Hence, even if great progresses have been made in the 2-dimensional setting, the method is not suited for general dimensions.

- Another possible generalization to the multivariate setting is based on the property 3. In particular a construction based on convolution led to the so called *Box-splines*. Again, even in the 2-dimensions the problem can be handle, for higher dimensions is still an open problem.

The property 2. is the motivation for a general framework in higher dimensions. This approach has allowed to develop a beautiful theory where *all space dimensions can be handled in the same way.* The resulting approximation spaces no longer consist of piecewise polynomials, so they can not be called splines. The new functions are known with the "fashionable words" of *Radial Basis Functions* (in what follows we refer to them simply as RBF).

1.2 From cubic splines to RBF

To get a better idea, let us remind that the set $\mathcal{S}_3(X)$ has the basis of truncated powers $(\cdot - x_j)_+^3$, $1 \le j \le N$ plus an arbitrary basis for $\mathbb{P}_3(\mathbb{R})$. Hence every $s \in \mathcal{N}_3(X)$ can be represented in the form

$$s(x) = \sum_{j=1}^N a_j (x - x_j)_+^3 + \sum_{j=0}^3 b_j x^j, \quad x \in [a, b]\,. \tag{1.4}$$

Because s is a *natural spline* we have the additional information that s is linear on the two outer intervals. That is on $[a, x_1]$ the spline is simply $s(x) = b_0 + b_1 x$ (since $b_2 = b_3 = 0$). Thus (1.4) becomes

$$s(x) = \sum_{j=1}^N a_j (x - x_j)_+^3 + b_0 + b_1 x, \quad x \in [a, x_1]\,. \tag{1.5}$$

To derive the representation of s in $[x_N, b]$ we have simply to remove all subscripts $+$ on the functions $(\cdot - x_j)^3_+$ in (1.5). Expanding these cubics and rearranging the sums we get

$$s(x) = \sum_{l=0}^{3} \binom{3}{l} (-1)^{3-l} \left(\sum_{j=1}^{N} a_j x_j^{3-l} \right) x^l + b_0 + b_1 x, \quad x \in [x_N, b]\,. \tag{1.6}$$

Thus, for s to be a natural spline, the coefficients of s have to satisfy

$$\sum_{j=1}^{N} a_j = \sum_{j=1}^{N} a_j x_j = 0\,. \tag{1.7}$$

This is a first characterization of natural cubic splines.

One more step. Using the identity $x^3_+ = \frac{(|x|^3 + x^3)}{2}$, and thanks to the relations (6.1.4) we get

$$\begin{aligned} s(x) &= \sum_{j=1}^{N} \frac{a_j}{2} |x - x_j|^3 + \sum_{j=1}^{N} \frac{a_j}{2} (x - x_j)^3 + b_0 + b_1 x \\ &= \sum_{j=1}^{N} \frac{a_j}{2} |x - x_j|^3 + \sum_{l=0}^{3} \frac{1}{2} \binom{3}{l} (-1)^{3-l} \left(\sum_{j=1}^{N} a_j x_j^{3-l} x^l \right) + b_0 + b_1 x \\ &= \sum_{j=1}^{N} \tilde{a}_j |x - x_j|^3 + \tilde{b}_0 + \tilde{b}_1 x\,, \end{aligned}$$

where $\tilde{a}_j = a_j/2, \;\; 1 \le j \le N$, $\tilde{b}_0 = b_0 - \frac{1}{2}\sum_{j=1}^{N} a_j x_j^3$ and $\tilde{b}_1 = b_1 + \frac{3}{2}\sum_{j=1}^{N} a_j x_j^2$.

Proposition 1. *Every natural spline s has the representation*

$$s(x) = \sum_{j=1}^{N} a_j \phi(|x - x_j|) + p(x), \quad x \in \mathbb{R} \tag{1.8}$$

where $\phi(r) = r^3$, $r \ge 0$ and $p \in \mathbb{P}_1(\mathbb{R})$. The coefficients $\{a_j\}$ have to satisfy the relations (6.1.4). On the contrary, for every set $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}$ of pairwise distinct points and for every $f \in \mathbb{R}^N$ there exists a function s of the form (6.1.4), with (6.1.4), that interpolates the data, i.e. $s(x_j) = f(x_j)$, $1 \le j \le N$.

This is the starting point for understanding the origin of RBF. The resulting interpolant is, up to a low-degree polynomial, a linear combination of *shifts* of a radial function $\Phi = \phi(|\cdot|)$. The function is then called *radial* because is the composition of a univariate function with the Euclidean norm on $\mathbb{R}$.

The generalization to $\mathbb{R}^d$ is straightforward where the name "radial" becomes even more evident. In fact

$$s(x) = \sum_{i=1}^{N} a_j \phi(\|x - x_j\|_2) + p(x), \;\; x \in \mathbb{R}^d \,, \tag{1.9}$$

where $\phi : [0, \infty) \to \mathbb{R}$ is a univariate fixed function and $p \in \mathbb{P}_{m-1}(\mathbb{R}^d)$ is a low degree d-variate polynomial. The additional conditions on the coefficients (corresponding to (6.1.4)) become

$$\sum_{i=1}^{N} a_j q(x_j) = 0, \;\; \forall\, q \in \mathbb{P}_{m-1}(\mathbb{R}^d)\,. \tag{1.10}$$

In many cases (see Lecture 2), we can avoid the *side conditions* on the coefficients (1.10). In these cases the interpolation problem has solution if the matrix

$$A_{\phi,X} := (\phi(\|x_i - x_j\|_2)_{1 \le i,j \le N} \,,$$

is invertible. To be more precise we ask

Problem 1. *Does there exist a function* $\phi : [0, \infty) \to \mathbb{R}$ *such that for all* $d, N \in \mathbb{N}$ *and all pairwise distrinct* $x_1, \ldots, x_n \in \mathbb{R}^d$ *the matrix* $A_{\phi,X}$ *is nonsingular ?*

The answer is affirmative. Examples of functions that allow to build matrices nonsingular are: the *gaussians* $\phi(r) = e^{-\alpha r^2}$, $\alpha > 0$, the *inverse multiquadric* $\phi(r) = (c^2 + r^2)^{-1/2}$ and the *multiquadric* $\phi(r) = (c^2 + r^2)^{1/2}$, $c > 0$. In the two first cases it is even true that the matrix $A_{\phi,X}$ is always positive definite (and so invertible).

Remark. In what follows, in the context of RBFs, instead of $A_{\phi,X}$ we shall use simply A thinking to the interpolation matrix with radial basis functions.

1.3 The scattered data interpolation problem

In many disciplines one faces the following problem: *we are given a set of data (measurements, locations at which these measurements are taken,...) and we want to find a rule which allows to get information about the process we are studying also at locations different from those at which the measurements are taken (or provided).*

The main reasons why we are interested on such a problem in our setting are:

- Scattered data fitting is a fundamental problem in approximation theory and data modeling in general
- Mathematical challenge: we want a *well-posed problem formulation*
- This will naturally lead to *distance matrices*
- Later we generalize to *radial basis functions* or *positive definite kernels*

Problem 2. *Given data $(\boldsymbol{x}_j, y_j)$, $j = 1, \dots, N$, with $\boldsymbol{x}_j \in \mathbb{R}^d$, $y_j \in \mathbb{R}$, find a (continuous) function $\mathcal{P}_f$ (depending on f) such that $\mathcal{P}_f(\boldsymbol{x}_j) = y_j$, $j = 1, \dots, N$.*

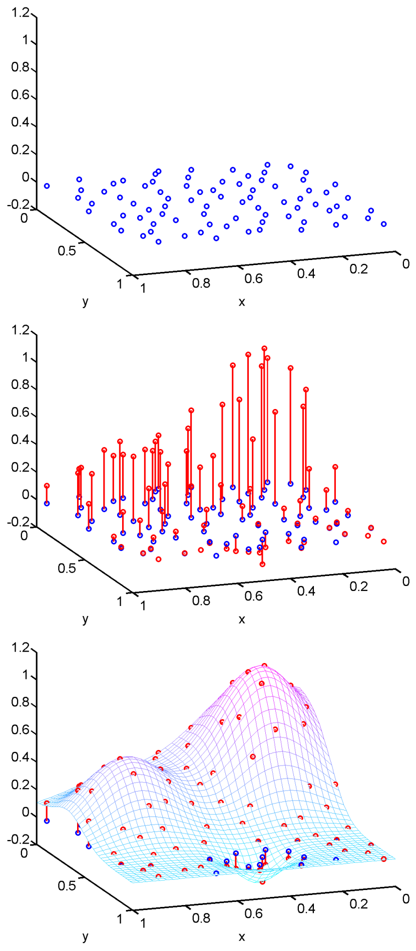


Figure 1.1: Data points, data values and data function

Now, assume $\mathcal{P}_f$ is a linear combination of certain *basis functions* B_k, that is

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{k=1}^{N} c_k B_k(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathbb{R}^d. \tag{1.11}$$

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

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

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

The scattered data fitting problem will be *well-posed*, that is a solution to the problem will exist and be unique, *if and only if the matrix A is non-singular*.

1.3.1 The Haar-Mairhuber-Curtis theorem

We need to introduce the following definition

Definition 1. *Let the finite-dimensional linear function space $\mathcal{B} \subset \mathcal{C}(\Omega)$ have a basis $\{B_1, \ldots, B_N\}$. Then $\mathcal{B}$ is a* **Haar space** *on Ω if*

$$\texttt{det}(A) \neq 0$$

for any set of distinct points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \Omega$. Here, A is the matrix with entries $A_{i,j} = B_j(\boldsymbol{x}_i)$.

The existence of a Haar space guarantees the invertibility of the matrix A. In the univariate setting it is well known that one can interpolate to arbitrary data at N distinct data sites using a polynomial of degree $N-1$. This means that the polynomials of degree $N-1$ form an N-dimensional Haar space for the set of distinct points $X = \{x_1, \ldots, x_N\}$.

This is a counterexample useful to understand the necessity of a different approach than using polynomials.

Example 1. It is not possible to perform unique interpolation with (multivariate) polynomials of degree N to data given at arbitrary locations in $\mathbb{R}^2$.

The Haar-Mairhuber-Curtis theorem tells us that if we want to have a well-posed multivariate scattered data interpolation problem **we can no longer fix in advance the set of basis functions we plan to use for interpolation of arbitrary scattered data**. Instead, the *basis should depend on the data points.*

Theorem 1. (**Haar-Mairhuber-Curtis**)
If $\Omega \subset \mathbb{R}^d$, $d \geq 2$ contains an interior point, then there exist no Haar spaces of continuous functions except for the 1-dimensional case.

Proof. Let $d \geq 2$ and assume that $\mathcal{B}$ is a Haar space with basis $\{B_1, \ldots, B_N\}$ with $N \geq 2$. We show that this leads to a contradiction. In fact, let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ be a set of N distinct points in $\Omega \subset \mathbb{R}^d$ and A the matrix such that $A_{j,k} = B_k(\boldsymbol{x}_j),\ j,k = 1, \ldots, N$. By the above definition of Haar space $\texttt{det}(A) \neq 0$. Now, consider the closed path P in Ω connecting only $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. *This is possibile since Ω by assumption contains an interior point.* We then can exchange $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ by moving them continuosly along P (without interfering with other point $\boldsymbol{x}_j$). This means that the rows 1 and 2 of the matrix A have been changed and so the determinant has changed sign. Since the determinant is a continuous function of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ we must have had $\texttt{det}(A) = 0$ at some point along P. This contradicts the fact that $\texttt{det}(A) \neq 0$. ■

1.3.2 Distance matrices

We want to construct a (continuous) function $\mathcal{P}_f$ that interpolates samples obtained from a test function f_d at data sites $\boldsymbol{x}_j \in [0,1]^d$. that is we want

$$\mathcal{P}_f(\boldsymbol{x}_j) = f_d(\boldsymbol{x}_j), \qquad \boldsymbol{x}_j \in [0,1]^d .$$

For example $f_d(\boldsymbol{x}) = 4^d \prod_{k=1}^{d} x_k(1-x_k)$, $\boldsymbol{x} = (x_1, \dots, x_d) \in [0,1]^d$, which is zeros on the boundary of the unit cube in $\mathbb{R}^d$ and has a maximum value of one at the center of the d-dimensional cube.

Assume for now that $\mathbf{d} = \mathbf{1}$. We have already seen that for small N one can use univariate polynomials; if N is relatively large it is better to use splines (the simplest approach is the C^0 piecewise linear splines "connect the dots")

A basis for the space of piecewise linear interpolating splines is

$$\{B_k = |\cdot - x_k| \;\; k = 1, \dots, N\} .$$

Hence our spline interpolant can be written as

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

and the coefficents c_k will be determined by the interpolation conditions

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

Some observations.

- The basis functions $B_k = |\cdot - x_k|$ are dependent on the data sites x_k as suggested by Haar-Mairhuber-Curtis.
- $B(x) = |x|$ is called *basic function.*
- The points x_k to which the basic function is shifted to form the basis functions, are usually referred to as *centers* or *knots.*
- Technically, one could choose these centers different from the data sites. However, usually *centers coincide with the data sites.* This simplifies the analysis of the method, and is sufficient for many applications. In fact, relatively little is known about the case when centers and data sites differ.
- B_k are (radially) symmetric about their centers x_k, $\longrightarrow$ **radial basis function**.

Now the coefficients c_k in the scattered data interpolation problem are found by solving the linear system

$$\begin{bmatrix} |x_1-x_1| & |x_1-x_2| & \dots & |x_1-x_N| \\ |x_2-x_1| & |x_2-x_2| & \dots & |x_2-x_N| \\ \vdots & \vdots & \ddots & \vdots \\ |x_N-x_1| & |x_N-x_2| & \dots & |x_N-x_N| \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} f_1(x_1) \\ f_1(x_2) \\ \vdots \\ f_1(x_N) \end{bmatrix} \tag{1.12}$$

- The matrix in (1.12) is an example of **distance matrix**.
- Distance matrices have been studied in geometry and analysis in the context of isometric embeddings of metric spaces for a long time.
- It is known that the distance matrix based on the Euclidean distance between a set of distinct points in $\mathbb{R}^d$ is always non-singular (see below).
- Therefore, **our scattered data interpolation problem is well-posed**.

Since distance matrices are non-singular for Euclidean distances *in any space dimension* d we have an immediate generalization: for the scattered data interpolation problem on $[0,1]^d$ we can take

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{k=1}^{N} c_k \|\boldsymbol{x}-\boldsymbol{x}_k\|_2, \qquad \boldsymbol{x} \in [0,1]^d, \tag{1.13}$$

and find the c_k by solving

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

- Note that the basis is again data dependent
- Piecewise linear splines in higher space dimensions are usually constructed differently (via a cardinal basis on an underlying computational mesh)
- For $d > 1$ the space $\texttt{span}\{\|\cdot-\boldsymbol{x}_k\|_2,\ k=1,\dots,N\}$ is **not** the same as piecewise linear splines

In order to show the non-singularity of our distance matrices we use the Courant-Fischer theorem (see for example the book by Meyer [58]):

Theorem 2. *Let A be a real symmetric $N \times N$ matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$, then*

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

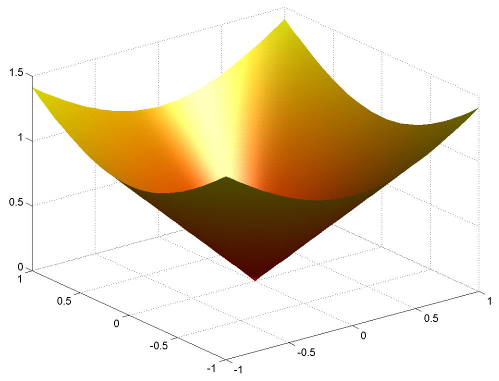


Figure 1.2: A typical basis function for the Euclidean distance matrix fit, $B_k(\boldsymbol{x}) = \|\boldsymbol{x}-\boldsymbol{x}_k\|_2$ with $\boldsymbol{x}_k = \mathbf{0}$ and $d = 2$.

Definition 2. *A real symmetric matrix A is called* **Conditionally Negative Definite (CND)** *of order one (or* almost negative definite*) if its associated quadratic form is negative, that is*

$$\sum_{j=1}^{N}\sum_{k=1}^{N} c_j c_k A_{jk} < 0 \tag{1.14}$$

for all $\boldsymbol{c} = [c_1, \ldots, c_N]^T \neq \mathbf{0} \in \mathbb{R}^N$ that satisfy $\displaystyle\sum_{j=1}^{N} c_j = 0$.

A matrix which is CND of order 1, is CND on a subspace of dimension $N-1$, that is it has at least $N-1$ negative eigenvalues. Indeed holds the following result.

Theorem 3. *An $N \times N$ symmetric matrix A which is almost negative definite and has a non-negative trace possesses one positive and $N-1$ negative eigenvalues.*

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

$$\lambda_2 = \min_{\dim\mathcal{V}=N-1} \max_{\substack{\boldsymbol{x}\in\mathcal{V}\\ \|\boldsymbol{x}\|=1}} \boldsymbol{x}^T A\boldsymbol{x} \leq \max_{\substack{\boldsymbol{c}:\ \sum c_k=0\\ \|\boldsymbol{c}\|=1}} \boldsymbol{c}^T A\boldsymbol{c} < 0,$$

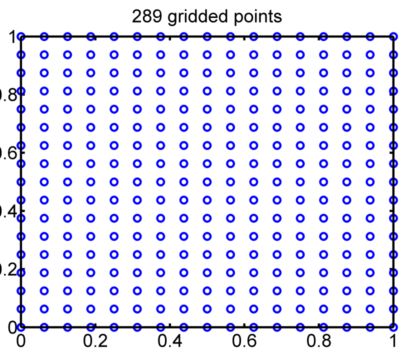
so that A has at least $N-1$ negative eigenvalues. But since $\mathrm{tr}(A) = \displaystyle\sum_{k=1}^{N} \lambda_k \geq 0$, A also must have at least one positive eigenvalue. ■

Example 2. It is known that $\phi(r) = r$ is a *strictly conditionally negative definite function* of order one, i.e., the matrix A with $A_{jk} = \|\boldsymbol{x}_j - \boldsymbol{x}_k\|_2$ is almost negative definite. Moreover, since $A_{jj} = \phi(0) = 0$, $j = 1, \ldots, N$ then $\mathrm{tr}(A) = 0$. Therefore, our distance matrix is non-singular by the above theorem.

1.3.3 Data sets

Depending on the type of approximation problem we are given, we may or may not be able to select where the data is collected, i.e., the location of the data sites or *design*.

Standard choices in low space dimensions are depicted in Fig. 1.3. In *higher space dimensions* it is important to have space-filling (or low-discrepancy) quasi-random point sets. Examples include: **Halton points, Sobol points, lattice designs, Latin hypercube designs and quite a few others (digital nets, Faure, Niederreiter, etc.)**.



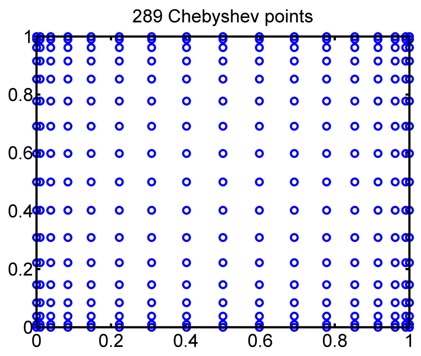


Figure 1.3: tensor products of equally spaced points and tensor products of Chebyshev points

Definition 3. *Given a sequence $X = \{x_1, ..., x_N\}$ its discrepancy is*

$$D_N(X) := \sup_{B\in J} \left| \frac{\#(X,B)}{N} - \lambda_d(B) \right| \tag{1.15}$$

where

- $J := \prod_{i=1}^d [a_i, b_i) = \{x \in \mathbb{R}^d : \ a_i \le x_i \le b_i, \ 0 \le a_i < b_i < 1\}$ *(d-dimensional intervals),*
- $\#(X, B)$ *is the number of points of X in B.*
- λ_d *is Lebesgue measure*

When $D_N(X) \approx \texttt{mis}(B)$ then D_N is called **low discrepancy**.

Low-discrepancy sequences are known also as *quasi-random* sequences, because they are often used as uniformely distributed random numbers.

Example 3. A typical application of such sequences is `numerical quadrature (or cubature)`. For example in the one-dimensional case

$$\int_0^1 f(t)dt \approx \frac{1}{N}\sum_{i=1}^N f(x_i)\,. \tag{1.16}$$

- If in (1.16) $x_i = i/N$ then we get the rectangle formula.
- If x_i are random numbers, then (1.16) is the Montecarlo method.
- If x_i are a low discrepancy sequences then (1.16) is a quasi-Montecarlo method.

1.3.4 Halton points

Here we show how one can generate `Halton points` in every spatial dimension. Halton points are uniformely distributed random point in $(0,1)^d$ generated from `Van der Corput` sequences. We start by generating Van der Corput sequences. Let $k \in \mathbb{N}$ be chosen.

(i) Every $n \in \mathbb{N}$ can be written as

$$n = \sum_{i=0}^{k} a_i p^i$$

where the coefficients a_i are integer such that $0 \le a_i < p$. For example taking $n = 10$ and $p = 3$

$$10 = 1 \cdot 3^0 + 0 \cdot 3^1 + 1 \cdot 3^2$$

giving $k = 2, a_0 = a_2 = 1,\ a_1 = 0$.

(ii) We define the function $h_p : \mathbb{N} \to [0,1)$ as $h_p(n) = \frac{\sum_{i=0}^{k} a_i}{p^{i+1}}$. For example

$$h_p(10) = \frac{1}{3} + \frac{1}{3^3} = \frac{10}{27}\,.$$

(iii) The Van der Corput sequence is then

$$h_{p,N} = \{h_p(n)\, :\, n = 0, 1, \ldots, N\}\,.$$

In our example

$$h_{3,10} = \{0, \frac{1}{3}, \frac{2}{3}, \frac{1}{9}, \frac{4}{9}, \frac{7}{9}, \frac{2}{9}, \frac{5}{9}, \frac{8}{9}, \frac{1}{27}, \frac{10}{27}\}\,.$$

Starting from the Van der Corput sequence, the Halton seqnuence is generated as follows: **take d distinct primes $p_1, \ldots, p_d$ and generated** $h_{p_1,N}, \ldots, h_{p_d,N}$ that we use as coordinates of d-dimensional points, so that

$$H_{d,N} = \{(h_{p_1,N}(n), \ldots, h_{p_d,N}(n)\, :\, n = 0, \ldots, N\}$$

is the set of $N+1$ Halton points in $[0,1)^d$.

Proposition 2. *Halton points form a nested sequence, that is if $M < N$ then $H_{d,M} \subset H_{d,N}$.*

These points can be constructed sequentially. Similar to these points are **Leja sequences** [24]. As a final observation, for Halton points we have

$$D_N(H_{d,N}) \leq \frac{C(\log N)^d}{N} \,.$$

$\diamond\diamond$

1. In Matlab the program `haltonseq.m` by Daniel Dougherty, dowloadable at the **Matlab Central File Exchange**, generates Halton points in every space dimension. The call is `haltonseq(numpts, dim)`. Notice that in this implementation the point $\mathbf{0} = (0, ..., 0)^T$ is not part of the point set, that is they are generated starting from $n = 1$ instead of $n = 0$ as described above. In recent Matlab versions, the function `P=haltonset(n,d)` computes `n` Halton points in dimension `d`.

2. Analogously, Sobol points can be generated in Matlab by `P=sobolset(d)`. This call constructs a d-dimensional point set `P` of the sobolset class, with default property settings. *For example*, if $d = 2$, `P` is an array *intmax* $\times$ 2. If one wishes different properties, the call become `P=sobolset(d, p1, var1, p2,var2, ...)` that specifies property name/value pairs used to construct `P`.

3. Similarly, the Matlab function `X = lhsdesign(n,p)` returns an $n \times p$ matrix, `X`, containing a latin hypercube sample of n values on each of p variables. For each column of `X`, the n values are randomly distributed with one from each interval $(0, 1/n), (1/n, 2/n), ..., (1 - 1/n, 1)$, and they are randomly permuted.

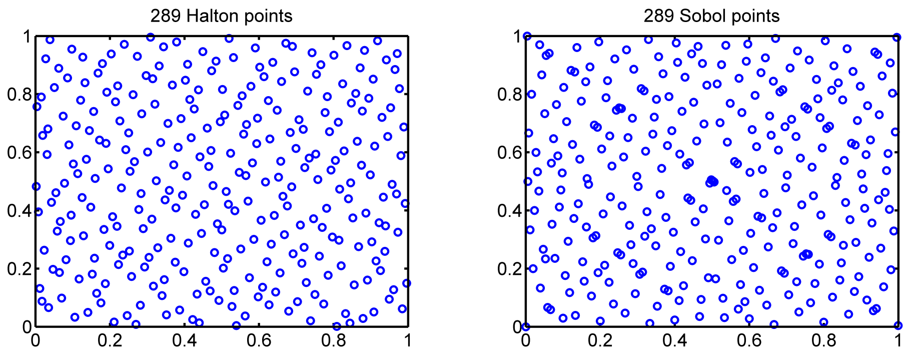


Figure 1.4: Halton and Sobol points

The difference between the standard (tensor product) designs and the quasi-random designs shows especially in higher space dimensions, as shown in Fig. 1.6

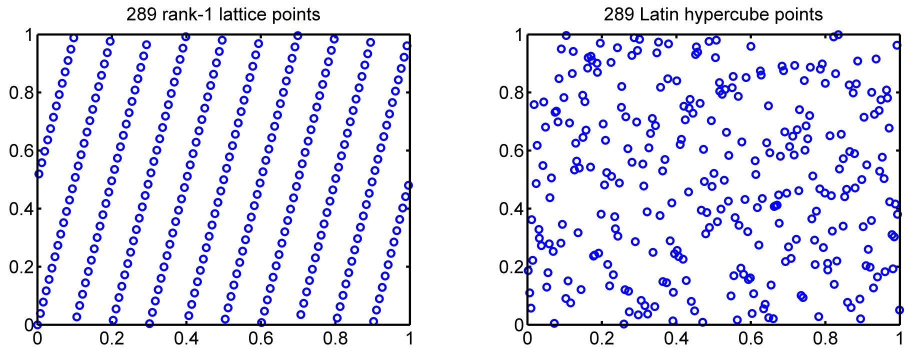


Figure 1.5: Lattice and Latin points

1.4 Exercises

The Haar-Maierhuber-Curtis theorem told us that is not possibile to interpolate by multivaraite polynomials of degree $N \geq 2$ scattered data in dimension $d \geq 2$. This suggested to take a basis of function localized at the so called *data-sites* .

The tools that we need for these exercises are

- The *Halton points* on the hypercube $[0,1]^d$, $d \geq 1$.
- The `fill-distance` (or *mesh size*) $h_{X,\Omega}$ of a set $X \subset \Omega$ con $\Omega \subseteq \mathbb{R}^d$ that is

$$h_{X,\Omega} = \sup_{x \in \Omega} \min_{\mathbf{x}_j \in X} \|x - x_j\|_2 \,, \tag{1.17}$$

  In Matlab, this distance can be easily computed by the command `hX=max(min(DME'))`, where `DME` is the distance matrix, generated by the function `DistanceMatrix.m` evaluated at a set of `target (or evaluation) points` (for example an equispaced grid finer than the of the data-sites `X`).

- The functions to be approximated are

$$\begin{aligned} f_s(\mathbf{x}) &= 4^s \prod_{k=1}^{s} x_k(1-x_k), \quad \mathbf{x} = (x_1, \ldots, x_s) \in [0,1]^d & (1.18)\\ \mathtt{sinc}(\mathbf{x}) &= \prod_{k=1}^{s} \frac{\sin(\pi x_k)}{\pi x_k}\,. & (1.19)\end{aligned}$$

1. By means of the Matlab function `haltonseq.m` compute $N = 5^d$ Halton points in dimensions $d = 1, 2, 3$. Compute for each set the corrisponding `fill-distance`, $h_{X,\Omega}$.
2. Verify graphically the *nested property* of Halton points, that is

$$H_{d,M} \subset H_{d,N}, \quad M < N\,. \tag{1.20}$$

In practice, using different colours plot the Halton points for different values of M and N.

3. Again, by using the function `DistanceMatrix.m` on different set of Halton points of dimension $d = 2$, verify that the corresponding *distance-matrix*, say `A`, is ill-conditioned, by computing its condtion number in the 2-norm (in Matlab `cond(A)`).

4. In dimension $d = 2$, using the function `DistanceMatrixFit.m`, build the RBF interpolant using the basis $\phi_k(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_k\|_2$ (that is, the translates at $\boldsymbol{x}_k$ of the basic function $\phi(r) = r$) of the functions (1.18) and (1.19) by computing the Root Mean Square Error, RMSE (see for its definition Lecture 4). Verify that as N increases, the error decreases. Notice: the RMSE has to be evaluated in a finer grid of evaluation points.

5. Repeat the previous exercise by using the Gaussian radial basis function, $\Phi(\boldsymbol{x}) = \mathrm{e}^{-\epsilon^2\|\boldsymbol{x}\|^2}$, $\epsilon > 0$ again for $d = 2$. For accomplish this interpolation, use the function `RBFInterpolation2D.m` that generalizes the `DistanceMatrixFit.m`.

The Matlab files can be downloaded at the link

http://www.math.unipd.it/~demarchi/TAA2010

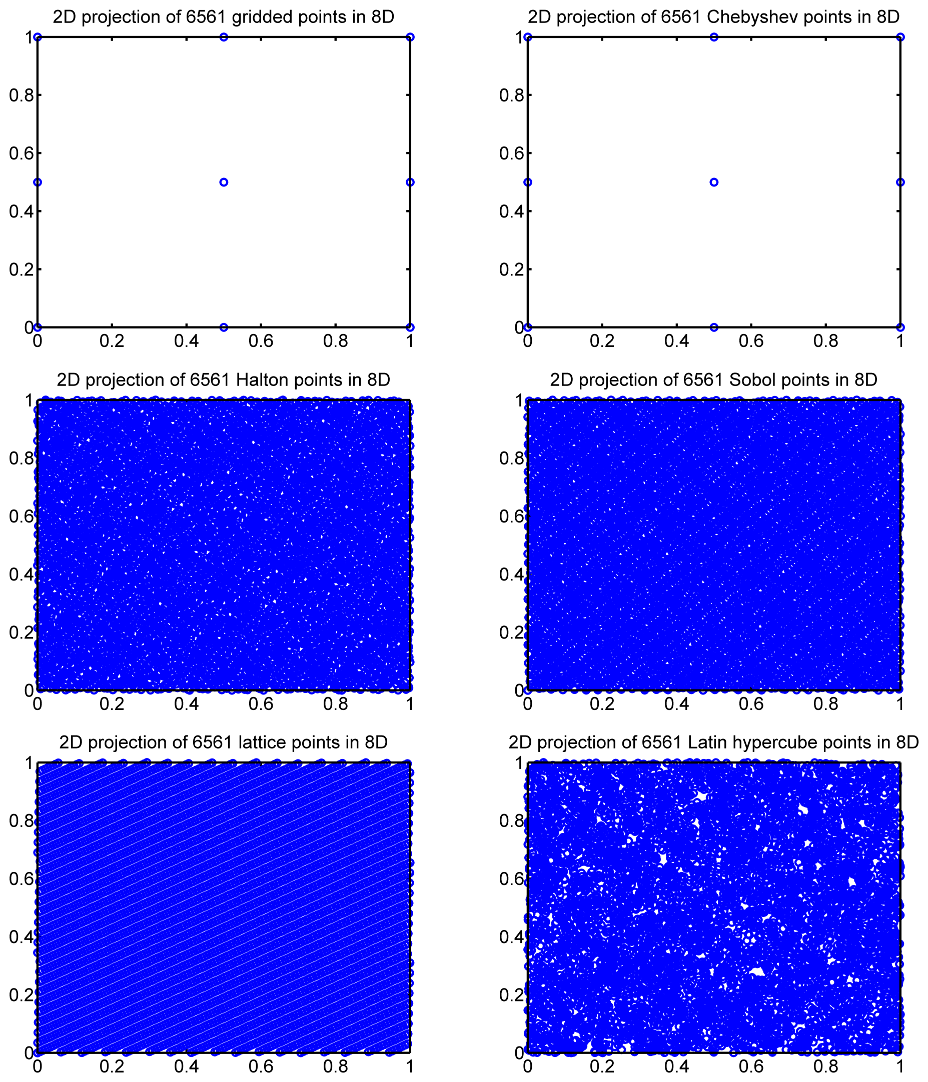


Figure 1.6: Projections in 2D of different point sets

Lecture 2

Positive definite functions

In the first lecture we saw that the scattered data interpolation problem with RBFs leads to the solution of a linear system

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

with $A_{i,j} = \phi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2)$ and y_i the i-th data value. The solution of the system requires that the matrix A is non-singular. The situation is favourable if we know in advance that the matrix is *positive definite*. Moroever we would like to characterize the class of functions ϕ for which the matrix is positive definite.

2.1 Positive definite matrices and functions

We start by the definition of *positive definite matrices.*

Definition 4. *A real symmetric matrix A is called* `positive semi-definite` *if its associated quadratic form $\mathbf{c}^T A\boldsymbol{c} \geq 0$, that is*

$$\sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j A_{i,j} \geq 0 \tag{2.1}$$

for $\boldsymbol{c} \in \mathbb{R}^N$. If the quadratic form (2.1) is zero only for $\boldsymbol{c} = \mathbf{0}$ then A is called `positive definite` *(sometimes we will use the shorthand notation PD).*

The most important property of such matrices, is that their eigenvalues are positive and so is its determinant. The contrary is not true, that is a matrix with positive determinant is not necessarly positive definite. Just consider as a trivial example the matrix (in Matlab notation) `A=[-1 0 ; 0 -3]` whose determinant is positive, but `A` is not PD.

Hence, if in (1.11) the basis B_k generates a positive definite interpolation matrix that we would always have a well-defined interpolation problem. In order to get such property, we need to introduce the class of *positive definite functions.*

Definition 5. *A continuous complex valued function $\Phi : \mathbb{R}^d \to \mathbb{C}$ is called* `positive semi-definite` *if, for all $N \in \mathbb{N}$, all sets of pairwise distinct points $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^d$ and $\boldsymbol{c} \in \mathbb{C}^N$ the quadratic form*

$$\sum_{i=1}^{N}\sum_{j=1}^{N} c_i \bar{c}_j \Phi(\boldsymbol{x}_i - \boldsymbol{x}_j) \tag{2.2}$$

is nonnegative. The function Φ is then called `positive definite` *if the quadratic form above is positive for $\boldsymbol{c} \in \mathbb{C}^N, \boldsymbol{c} \neq \boldsymbol{0}$.*

Example 4. The function $\Phi(\boldsymbol{x}) = e^{i\boldsymbol{x}^T\boldsymbol{y}}$, for a fixed $\boldsymbol{y} \in \mathbb{R}^d$, is PD on $\mathbb{R}^d$. If fact,

$$\begin{aligned}
\sum_{i=1}^{N}\sum_{j=1}^{N} c_i \bar{c}_j \Phi(\boldsymbol{x}_i - \boldsymbol{x}_j) &= \sum_{i=1}^{N}\sum_{j=1}^{N} c_i \bar{c}_j e^{i(\boldsymbol{x}_i - \boldsymbol{x}_j)^T\boldsymbol{y}} \\
&= \left(\sum_{i=1}^{N} c_i e^{i\boldsymbol{x}_i^T\boldsymbol{y}}\right)\left(\sum_{j=1}^{N} \bar{c}_j e^{-i\boldsymbol{x}_j^T\boldsymbol{y}}\right) \\
&= \left|\sum_{i=1}^{N} c_i e^{i\boldsymbol{x}_i^T\boldsymbol{y}}\right|^2 \geq 0.
\end{aligned}$$

Remark. From the previous definition and the discussion done on Lecture 1, we should use PD functions as basis, i.e. $B_i(\boldsymbol{x}) = \Phi(\boldsymbol{x} - \boldsymbol{x}_i)$, that is an interpolant of the form $P_f(\boldsymbol{x}) = \sum_{i=1}^{N} c_i B_i(\boldsymbol{x})$. Moreover at this point, we do not need P_f be a *radial* function, but simply *translation invariant* (that is, P_f is the same as the translated interpolant to the original data). We will characterize PD and radial functions in $\mathbb{R}^d$ later in this lecture.

⋄⋄

These functions have some properties that we summarize in the following theorem.

Theorem 4. *Suppose Φ is a positive semi-definite function. Then*

1. $\Phi(\boldsymbol{0}) \geq 0$ *and* $\Phi(\boldsymbol{0}) = 0$ *iff* $\Phi \equiv 0$.
2. $\Phi(-\boldsymbol{x}) = \overline{\Phi(\boldsymbol{x})}$ *for all* $\boldsymbol{x} \in \mathbb{R}^d$.
3. $|\Phi(\boldsymbol{x})| \leq \Phi(\boldsymbol{0})$ *for all* $\boldsymbol{x} \in \mathbb{R}^d$ *(boundness)*

4. *If $\{\Phi_i, i = 1, \ldots, m\}$ are positive semi-definite and $\alpha_j \geq 0$, $j = 1, ..., m$, then $\Phi = \sum_{i=1}^m \alpha_j \Phi_j$ is also positive semi-definite. If one of the Φ_j is positive definite and the corresponding coefficient α_j is positive, then Φ is also positive definite.*

5. *The product of two positive definite functions is positive definite.*

Proof. The proof can be found in [74, p. 65-66] or [32, 29-30]. ■

Remark. From property 2. of Theorem 4, it is clear that a positive semi-definite function is real valued iff is `even`. But we can also restrict to real coefficient vectors $\boldsymbol{c} \in \mathbb{R}^N$ in the quadratic form. In fact holds the following theorem.

Theorem 5. *Suppose $\Phi : \mathbb{R}^d \to \mathbb{R}$ is continuous. Then Φ is positive definite if and only if Φ is even and we have, for all $N \in \mathbb{N}$ and all $\boldsymbol{c} \in \mathbb{R} \backslash \{\boldsymbol{0}\}$ and all pairwise distinct $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$*

$$\sum_{i=1}^N \sum_{j=1}^N c_i c_j \Phi(\boldsymbol{x}_i - \boldsymbol{x}_j) > 0\,.$$

Proof. If Φ is PD and real valued, then it is even by the previous theorem (by property 2.). Letting $c_k = a_k + i\, b_k$ then

$$\begin{aligned}\sum_{i,j=1}^N c_i \bar{c}_j \Phi(\boldsymbol{x}_i - \boldsymbol{x}_j) &= \sum_{i,j=1}^N (a_i a_j + b_i b_j)\Phi(\boldsymbol{x}_i - \boldsymbol{x}_j) + \\ &+ i \sum_{i,j=1}^N a_j b_i [\Phi(\boldsymbol{x}_i - \boldsymbol{x}_j) - \Phi(\boldsymbol{x}_j - \boldsymbol{x}_i)]\,.\end{aligned}$$

As Φ is even, the second sum on the right hand side is zero. The first sum is nonnegative bacause of the assumption, vanishing only if $a_i = b_i = 0$. ■

Example 5. The cosine function is PD on $\mathbb{R}$. In fact, for all $x \in \mathbb{R}$, $\cos x = \frac{e^{ix} + e^{-ix}}{2}$. By property 4. of Theorem 4 and the fact that the exponential is PD (see Example 4), we conclude.

When we are dealing with `radial functions` i.e $\Phi(\boldsymbol{x}) = \phi(\|\boldsymbol{x}\|)$, then it will be convenient to refer to the *univariate* function ϕ as `positive definite radial function`. A consequence of this notational convention is the following Lemma.

Lemma 1. *If $\Phi(\boldsymbol{x}) = \phi(\|\boldsymbol{x}\|)$ is PD (or positive semi-definite) and radial on $\mathbb{R}^d$, then Φ is also PD (or positive semi-definite) and radial on $\mathbb{R}^\delta$ for any $\delta \leq d$.*

◇◇

The Schoenberg characterization of PD and radial functions

The classical way to characterize PD and radial functions is by this Theorem, due to Schoenberg [66].

Theorem 6. *A continuous function $\phi : [0,\infty) \to \mathbb{R}$ is PD and radial on $\mathbb{R}^d$ if and only if it is the Bessel transform of a finite nonnegative Borel measure μ on $[0,\infty)$, i.e.*

$$\phi(r) = \int_0^\infty \Omega_d(rt)d\mu(t)$$

where

$$\Omega_d(r) = \begin{cases} \cos(r) & d{=}1 \\ \Gamma(\frac{d}{2})\left(\frac{2}{r}\right)^{(d-2)/2} J_{(d-2)/2}(r) & d \ge 2 \end{cases}$$

Here J_ν is the classical Bessel function of the first kind of order ν, that is the solution of the Bessel differential equation. For example, around $x = 0$, this function can be expressed as the Taylor series

$$J_\nu(x) = \sum_{k=0}^{\infty} \frac{(-1)^k}{k!\,\Gamma(k+\nu+1)} \left(\frac{x}{2}\right)^{2k+\nu} .$$

In particular, as already seen, $\Phi(x) = \cos(x)$ can be seen as a fundamental PD and radial function on $\mathbb{R}$.

Observation. Since these lectures are intended for people with no preperation and mathematical background on measure theory and functional analysis, we prefer to avoid to introduce more results characterizing PD functions by using (Fourier) transforms. We prefer to use another and more comprehensible approach based on the definition of completely monotone and multiply monotone functions. Interested readers can satisfy their hunger for knowledge by looking at the books [8, 32, 74].

2.1.1 Completely monotone functions

In the framed Theorem 6 and successive remark, we observed that we avoid to characterize the positive definiteness of RBF using Fourier transforms, also because Fourier transforms are not always easy to compute.

Definition 6. *A function $\phi : [0,\infty) \to \mathbb{R}$ that is $\mathcal{C}[0,\infty) \cap \mathcal{C}^\infty(0,\infty)$ and satisfies*

$$(-1)^k \phi^{(k)}(r) \ge 0, \;\; r > 0,\; k = 0, 1, 2, \ldots$$

is called `Completely Monotone` *(shortly CM).*

This definition allows to verify when a function ϕ is positive definite and radial for all dimensions d.

Here we enumerate some of the most important PD functions showing that they are CM.

1. The function $\phi(r) = c,\ c \geq 0$ is CM on $[0, \infty)$.
2. The function $\phi(r) = e^{-c\,r},\ c \geq 0$ is CM on $[0, \infty)$ since
$$(-1)^k \phi^{(k)}(r) = c^k e^{-cr} \geq 0,\ k = 0, 1, 2, ...$$
3. The function $\phi(r) = \frac{1}{(1+r)^\beta},\ \beta \geq 0$ is CM on $[0, \infty)$ since
$$(-1)^k \phi^{(k)}(r) = (-1)^{2k} \beta(\beta-1) \cdots (\beta+k-1)(1+r)^{-\beta-k} \geq 0,\ k = 0, 1, 2, ...$$

There are now two Theorems to quote that give more informations

Theorem 7. *A function ϕ is CM on $[0, \infty)$ if and only of $\Phi = \phi(\|\cdot\|^2)$ is positive semi-definite and radial on $\mathbb{R}^d$ for all d.*

Notice that now Φ is defined via the *square* of the norm.

Theorem 8. *A function ϕ is CM on $[0, \infty)$ but not constant if and only of $\Phi = \phi(\|\cdot\|^2)$ is PD and radial on $\mathbb{R}^d$ for all d.*

Hence as a consequence of these two theorems, the functions $\phi(r) = e^{-cr}, c > 0$ and $\phi(r) = \frac{1}{(1+r)^\beta},\ \beta \geq 0$ are CM on $[0, \infty)$ and since they are not constant, in $\mathbb{R}^d$ for all d, we have

1. $\Phi(\boldsymbol{x}) = e^{-c^2\|\boldsymbol{x}\|^2},\ c > 0$ is positive definite and radial on $\mathbb{R}^d$ for all d. This is the family of `Gaussians`. The parameter c is a *shape parameter* that change the shape of the function, making it more spiky when $c \to \infty$ and flatter when $c \to 0$.
2. $\Phi(\boldsymbol{x}) = (1 + \|\boldsymbol{x}\|^2)^{-\beta},\ \beta \geq 0$ is positive definite and radial on $\mathbb{R}^d$ for all d. This is the family of `inverse multiquadrics`.

2.1.2 Multiply monotone functions

This characterization allows to check when a function is PD and radial on $\mathbb{R}^d$ for `some fixed` d.

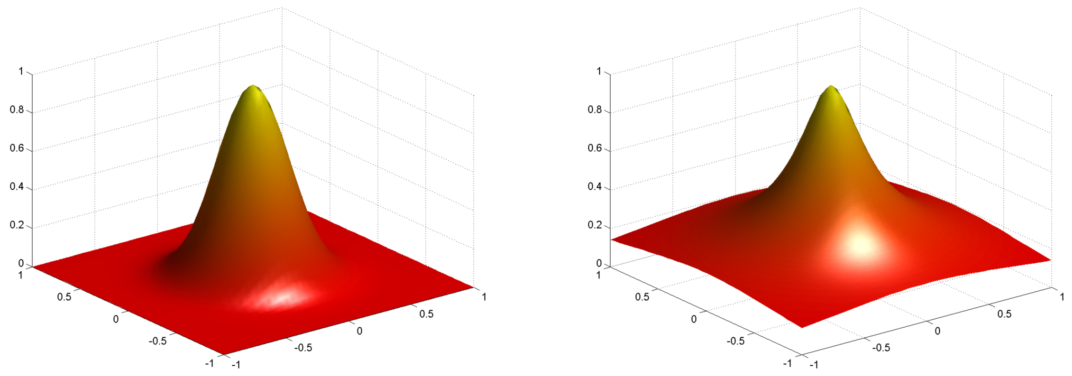


Figure 2.1: Left: Gaussian with $c = 3$. Right: inverse multiquadric with $\beta = 1$.

Definition 7. *A function $\phi : (0,\infty) \to \mathbb{R}$ which is $\mathcal{C}^{s-2}(0,\infty)$, $s \geq 0$ and for which $(-1)^k \phi^{(k)}(r) \geq 0$, non-increasing and convex for $k = 0, 1, \ldots, s-2$ is called* `s times monotone` *on $(0,\infty)$. In case $s = 1$ we only require $\phi \in \mathcal{C}(0,\infty)$ to be non-negative and non-increasing.*

In what follows sometimes we use the shorthand notation `MM` for multiply monotone functions.

Convexity can be expressed by the inequality $\phi'' \geq 0$ if ϕ'' exists. Hence, a MM function is essentially a CM function with *truncated monotonicity*.

Example 6. Here we introduce two `new` families of radial basis functions.

1. The `truncated power function`

$$\phi_k(r) = (1-r)_+^k$$

is k-times monotone for any k. Indeed

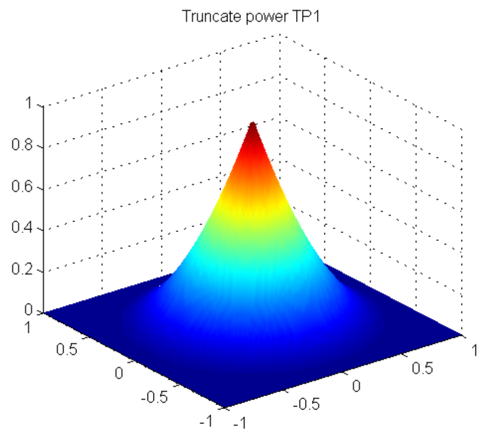
$$(-1)^s \phi_k^{(s)} = k(k-1)\cdots(k-s+1)(1-r)_+^{k-s} \geq 0, \quad s = 0, 1, \ldots, k$$

These functions (we will see later) lead to radial functions that are PD on $\mathbb{R}^d$ provided $k \geq \lfloor d/2 \rfloor + 1$.

2. Consider the integral operator I defined as

$$(If)(r) = \int_r^\infty f(t)dt\,, \quad r \geq 0\,,$$

with an f which is k-times monotone. Then If is $(k+1)$-times monotone. This follows from the fundamental theorem of calculus. This operator I plays an important role in the construction of `compactly supported` radial basis functions (that for lack of time we will not discuss in these lectures).



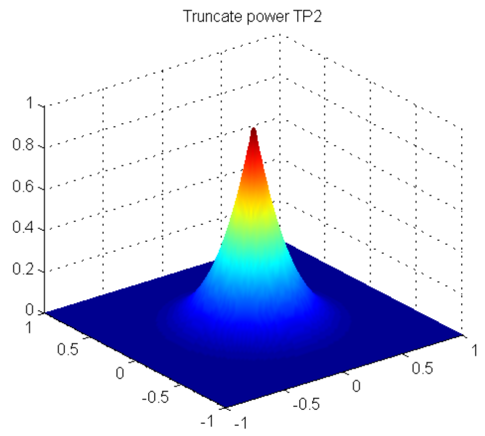


Figure 2.2: Left: truncated power with $k = 1$. Right: truncated power with $k = 2$.

2.1.3 Other positive definite radial functions

1. `Gaussians-Laguerre`. These are defined as

$$\Phi(\boldsymbol{x}) = e^{\|\boldsymbol{x}\|^2} L_n^{d/2}(\|\boldsymbol{x}\|^2)$$

where $L_n^{d/2}$ indicates the Laguerre polynomial of degree n and order $d/2$, that is

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

For example for $n = 1, 2$ and $d = 1, 2$ see Table 2.1.

d	$n = 1$	$n = 2$
1	$(3/2 - x^2)\mathrm{e}^{-x^2}$	$(15/8 - 5/2\,x^2 + 1/2x^4)\mathrm{e}^{-x^2}$
2	$(2 - \|\mathbf{x}\|^2)\mathrm{e}^{-\|\mathbf{x}\|^2}$	$(3 - 3\|\mathbf{x}\|^2 + 1/2\|\mathbf{x}\|^4)\mathrm{e}^{-\|\mathbf{x}\|^2}$

Table 2.1: Gaussians-Laguerre functions for different d and n

Notice, that the Gaussians-Laguerre, depends on the space dimension d. Therefore they are PD and radial on $\mathbb{R}^d$ (and therefore also on $\mathbb{R}^\delta$ for $\delta \le d$).

2. `Poisson` functions. These functions has the definition

$$\Phi(\boldsymbol{x}) = \frac{J_{d/2-1}(\|\boldsymbol{x}\|)}{\|\boldsymbol{x}\|^{d/2-1}}, \quad d \ge 2\,,$$

where J_p is the *Bessel function of the first kind and order* p. While these functions are not defined at the origin, they can be extended to be $\mathcal{C}^\infty(\mathbb{R}^d)$ (see Table 2.2 for

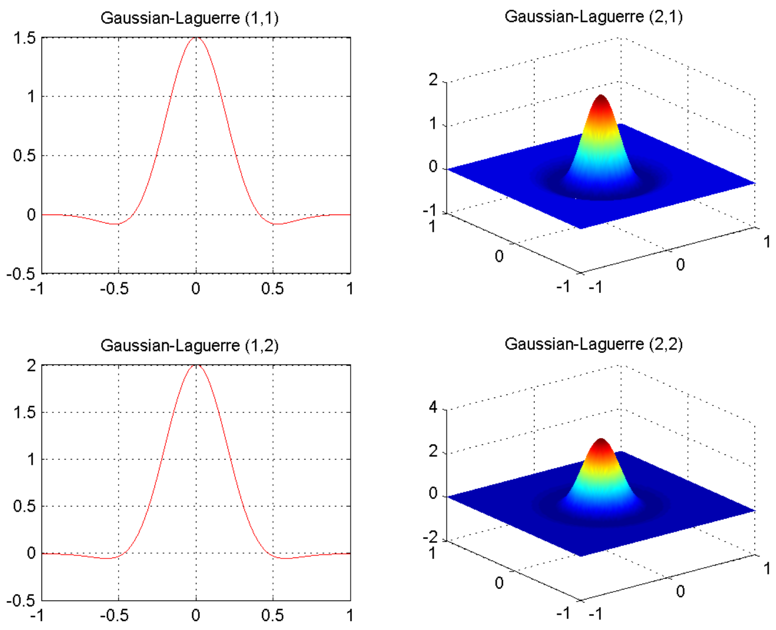


Figure 2.3: First row: G-L when $d = 1$ and $n = 1, 2$. Second row: G-L for $d = 2$ and $n = 1, 2$. In both cases we used the scaling factor $\epsilon = 3$. For the two dimensional case we also normalized so that $\Phi(\mathbf{0}) = 1$.

$d = 2$	$d = 3$	$d = 4$
$J_0(\Vert\mathbf{x}\Vert)$	$\sqrt{\frac{2}{\pi}}\frac{\sin(\Vert\mathbf{x}\Vert)}{\Vert\mathbf{x}\Vert}$	$\frac{J_1(\Vert\mathbf{x}\Vert)}{\Vert\mathbf{x}\Vert}$

Table 2.2: Poisson functions for various d

the cases $d = 2, 3, 4$.) **Notice:** J_p in Matlab can be computed using the function `besselj(p, z)` (where `z` is an array of evaluation points)

3. `Matérn` functions. They are defined as

$$\Phi(\boldsymbol{x}) = \frac{K_{\beta-d/2}(\Vert\boldsymbol{x}\Vert)\Vert\boldsymbol{x}\Vert^{\beta-d/2}}{2^{\beta-1}\Gamma(\beta)}, \quad d < 2\beta\,,$$

where K_p is the *modified Bessel function of the second kind* of order p, that can be defined as a function of the Bessel function of first kind as follows

$$K_p(x) = \frac{\pi}{2}\,\frac{J_{-p}(x) - J_p(x)}{\sin(\pi\, p)}$$

In Table 2.3 we present the Mátern functions for three values of β, indicated as

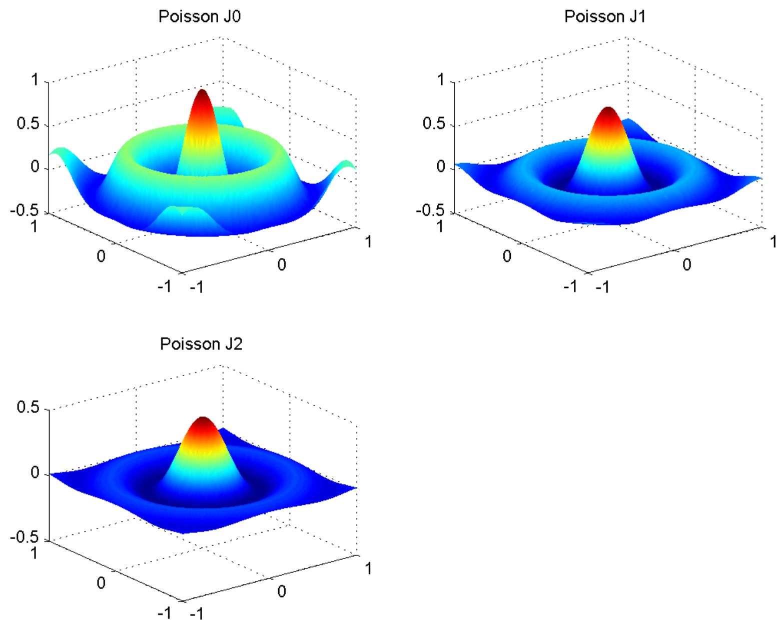


Figure 2.4: Poisson RBF for $d = 2, 3, 4$, respectively. Here the shape parameter is $\epsilon = 10$

$\beta_i,\ i = 1, 2, 3$. Notice that the *Matérn* function for β_1 is not differentiable at the origin; while for β_2 is $\mathcal{C}^2(\mathbb{R}^s)$ and for β_3 is $\mathcal{C}^4(\mathbb{R}^s)$.

$\beta_1 = \frac{d+1}{2}$	$\beta_2 = \frac{d+3}{2}$	$\beta_3 = \frac{d+5}{2}$
$\mathrm{e}^{-\Vert x\Vert}$	$(1 + \Vert x\Vert)\,\mathrm{e}^{-\Vert x\Vert}$	$\left(3 + 3\Vert x\Vert + \Vert x\Vert^2\right)\mathrm{e}^{-\Vert x\Vert}$

Table 2.3: Matérn functions for different values of the parameter β

4. The `generalized inverse multiquadrics`

$$\Phi(x) = (1 + \|x\|^2)^{-\beta}, \quad d < 2\beta$$

It is worth to mention that the requirement $d < 2\beta$ implies the dependence of the space dimension. This is required so that $\Phi(x) \in L_1(\mathbb{R}^d)$. As observed in [74, p. 77], this restriction can be relaxed and take $\beta > 0$ leading to a positive definite function in $\mathbb{R}^d$).

5. `Whittaker` functions. The idea of these functions is based on the following construction. Let $f \in \mathcal{C}[0, \infty)$ be a non-negative and not identically equal to zero, and define the function

$$\phi(r) = \int_0^\infty (1 - rt)_+^{k-1} f(t)dt\,. \tag{2.3}$$

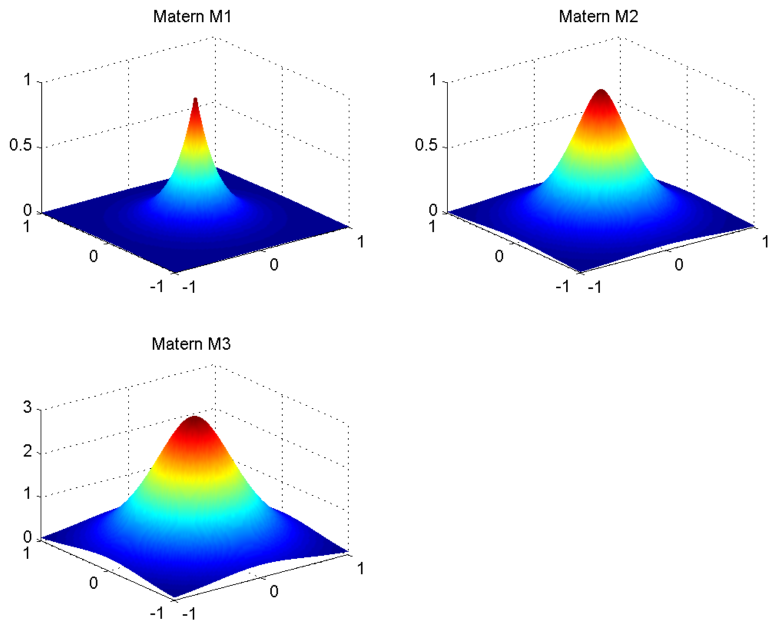


Figure 2.5: Matern RBF for different β, respectively. The shape parameter is $\epsilon = 5$

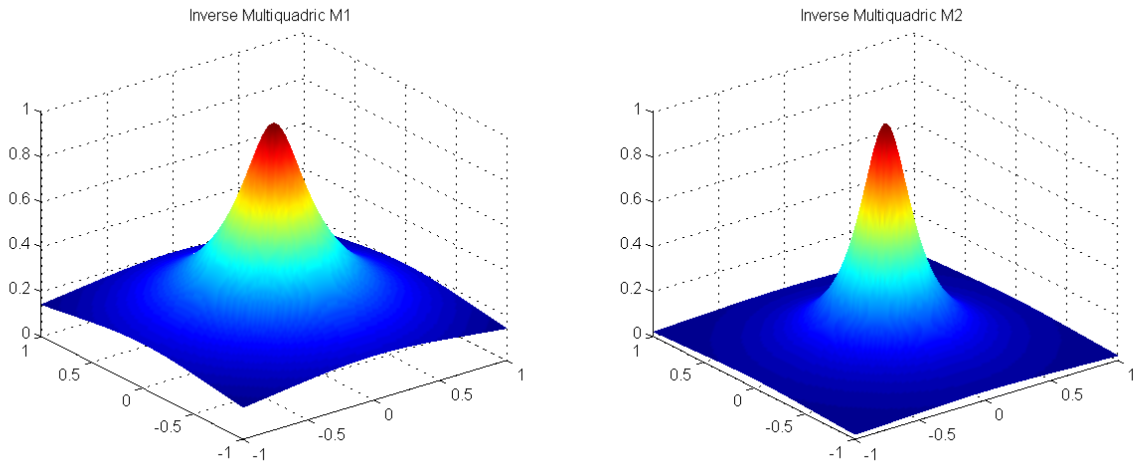


Figure 2.6: Inverse multiquadric with shape parameter $\epsilon = 3$. On the left $\beta = 1/2$ that corresponds to the *Hardy multiquadric.* On the right with $\beta = 1$ that corresponds to the *inverse quadric.*

Then the $\Phi(\boldsymbol{x}) = \phi(\|\boldsymbol{x}\|)$ is positive definite and radial on $\mathbb{R}^d$ provided $k \geq \lfloor \frac{d}{2} \rfloor + 2$.

In fact the quadratic form

$$\sum_{i,j=1}^{N} c_i c_j \phi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|) = \int_0^\infty \sum_{i,j=1}^{N} c_i c_j \varphi_{k-1}(t\|\boldsymbol{x}_i - \boldsymbol{x}_j\|) f(t) dt$$

which is non-negative since the truncated power $\varphi_{k-1}(\|\cdot\|)$ is positive semi-definite (see above) and f is non-negative. It is easy to see that it is indeed positive definite, and so ϕ is PD itself.

Then the choice of f gives different Whittaker functions. For example, as described in [32, p. 44], taking $f(t) = t^\alpha e^{\beta t}, \alpha \geq 0, \beta > 0$ we get a function with a long expression (see [32, p.44, formula (4.10)]) that uses the so called *Whittaker -M* function.

Special cases when $\beta = 1$ are provided in Table 2.4.

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

Table 2.4: Whittaker functions for various choices of k and α

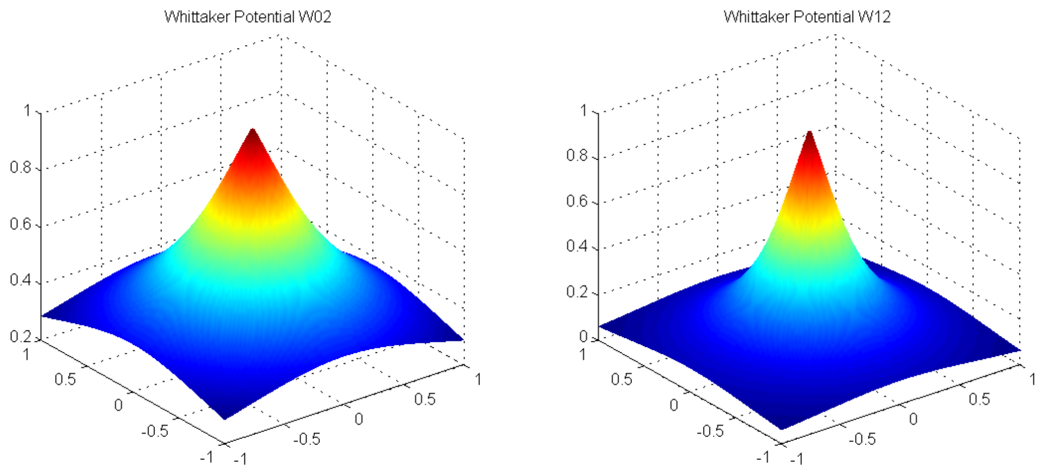


Figure 2.7: Whittaker functions with $\alpha = 0, k = 2$ and $\alpha = 1, k = 2$.

2.2 Exercises

1. Plot some of the radial function positive definite (centered in the origin). When $d = 1$ take $x \in [-1, 1]$ while for $d = 2$ consider $\mathbf{x} \in [-1, 1]^2$.

   - The *Gaussians -Laguerre* for $n = 1, 2$ and $d = 1, 2$. See Table 2.1 for the corresponding definitions.
   - The *Poisson* functions for $d = 2, 3, 4$ in $[-1, 1]^2$ using as shape parameter $\epsilon = 10$, see Table 2.2
   - The *Matérn* function in $[-1, 1]^2$, for three different values of β and shape parameter $\epsilon = 10$, as defined in Table 2.3.
   - The *generalized inverse multiquadrics* $\Phi(x) = (1 + \|x\|^2)^{-\beta}$, $s < 2\beta$, in $[-1, 1]^2$, in these two (simple) cases (with $\epsilon = 5$): $\beta = 1/2$ (which corresponds to the so-called *Hardy inverse multiquadrics*) and $\beta = 1$ (which is the *inverse quadrics*).
   - The *truncated powers* $\Phi(x) = (1 - \|x\|)_+^l$ for $l = 2, 4$ (in $[-1, 1]^2$).
   - The *Whittaker's potentials* in the square $[-1, 1]^2$ for different values of the parameters α, k and β, as defined in Table 2.4. For these last plots use $\beta = 1$.

2. Interpolate the *Franke function*

$$\begin{aligned} f(x_1, x_2) &= .75\exp[-((9x_1 - 2)^2 + (9x_2 - 2)^2)/4] + .75\exp[-(9x_1 + 1)^2/49 - (9x_2 + 1)/10] \\ &+ .5\exp[-((9x_1 - 7)^2 + (9x_2 - 3)^2)/4] - .2\exp[-(9x_1 - 4)^2 - (9x_2 - 7)^2]; \end{aligned}$$

   on a grid of 20×20 Chebyshev points in $[0, 1]^2$ with Poisson and Matérn functions. Compute also the corresponding RMSE.

Lecture 3

Conditionally positive definite functions

The multivariate polynomials are not suitable for solving the scattered data interpolation problem. Only data sites in special locations can guarantee well-posedness of the interpolation problem. In order to have a flavour of this setting, we introduce the notion of *unisolvency.*

Definition 8. *A set $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\} \subset \mathbb{R}^d$ is called m-`unisolvent`, if the only polynomial of total degree at most m interpolating the zero data on X is the zero polynomial.*

Similarly, X is unisolvent for $\mathbb{P}_m(\mathbb{R}^d)$ (i.e. polynomials in d variables of degree at most m) if there exists a unique polynomial in $\mathbb{P}_m(\mathbb{R}^d)$ of lowest possible degree which interpolates the data X.

Then, to get a unique solution of the polynomial interpolation problem with a d-variate polynomial of degree $\leq m$, to some given data, we need to find a subset of $X \subset \mathbb{R}^d$ with cardinality $M = \binom{m+d}{m} = \mathtt{dim}(\mathbb{P}_m(\mathbb{R}^d)$ which is m-unisolvent.

Example 7. These are examples of points that form unisolvent sets

- A simple example is this one. Take $N = 5$ points in $\mathbb{R}^2$. There is no unique way to use bivariate linear interpolation o quadratic. In fact linear polynomials have dimension $M = 3$, bivariate $M = 6$.
- The `Padua points` on the square $[-1, 1]^2$, form the first complete example of unisolvent points whose *Lebesgue constant* has optimal growth (cf. [6]).
- In $\mathbb{R}^2$ the `Chung and Yao points`. The construction of these points is based on lattices. In practise, a lattice $\Lambda \subset \mathbb{R}^d$ has the form $\Lambda = \left\{ \sum_{i=1}^{d} h_i v_i,\ h_i \in \mathbb{Z} \right\}$ with $\{v_i, \dots, v_d\}$ a basis for $\mathbb{R}^d$. For details see the paper [17].

The m-unisolvency of the set $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ is equivalent to the fact that the matrix P such that

$$P_{i,j} = p_j(x_i), \;\; i = 1, \ldots, N, \;\; j = 1, ..., M$$

for any polynomial basis p_j, has full (column)-rank. For $N = M$ this is the classical polynomial interpolation matrix.

This observation, can be easily checked when in $\mathbb{R}^2$ we take 3 collinear points: they are not 1-unisolvent, since a linear interpolant, that is a plane through three arbitrary heights at these three collinear points is not uniquely determined. Otherwise such a set is 1-unisolvent.

Remark. This problem arises when we want to construct interpolants with *polynomial precision.* That is, interpolants that reproduce polynomials.

Example 8. Take the function $f(x, y) = (x+y)/2$ on the unit square $[0, 1]^2$. Using Gaussian RBF (with $\epsilon = 6$) interpolate it on a grid of $N = 1089 = 33 \times 33$ uniformly distributed points. This will lead to an interpolant which is not a linear function as is f (i.e. the interpolation does not work).

Hence, instead of using an interpolant of the form $P_f(\boldsymbol{x}) = \sum_{k=1}^{N} c_k e^{-\epsilon^2 \|\boldsymbol{x}-\boldsymbol{x}_k\|^2}$, we use the following

$$P_f(\boldsymbol{x}) = \sum_{i=1}^{N} c_i e^{-\epsilon^2 \|\boldsymbol{x}-\boldsymbol{x}_i\|^2} + \underbrace{c_{N+1} + c_{N+2}x + c_{N+3}y}_{\text{polynomial part}} \,. \tag{3.1}$$

Having only N interpolation conditions, namely

$$P_f(\boldsymbol{x}_i) = (x_i + y_i)/2, \;\; i = 1, \ldots, N \,, \tag{3.2}$$

how can we find the remaining three conditions so that the resulting system will be square? As we shall see later, the solution is

$$\sum_{i=1}^{N} c_i = 0, \; \sum_{i=1}^{N} c_i x_i = 0, \; \sum_{i=1}^{N} c_i y_i = 0 \,. \tag{3.3}$$

Combining the interpolation conditions (3.2) with the *side conditions* (3.3), the resulting linear system becomes

$$\begin{bmatrix} A & P \\ P^T & O \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{0} \end{bmatrix} \tag{3.4}$$

where the matrix A and the vectors $\boldsymbol{c}$ and $\boldsymbol{y}$ are the same as in Lecture 1; while P is a $N \times 3$ matrix with entries $P_{i,j} = p_j(\boldsymbol{x}_i)$ with $p_1(\boldsymbol{x}) = 1, \; p_2(\boldsymbol{x}) = x, \; p_3(\boldsymbol{x}) = y$. Finally the matrix O is 3×3 of zeros.

The general form of the (3.2) is

$$P_f(\boldsymbol{x}) = \sum_{i=1}^{N} c_i \phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|) + \sum_{k=1}^{M} d_k p_k(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^d. \tag{3.5}$$

where $p_1, \ldots, p_M$ is a basis for the d-variate polynomials of degree $\le m-1$, whose dimension is $M = \binom{m-1+d}{m-1}$. The side-conditions become

$$\sum_{i=1}^{N} c_i p_j(\boldsymbol{x}_i) = 0, \quad j = 1, \ldots, M, \tag{3.6}$$

ensuring a unique solution for the system

$$\begin{bmatrix} A & P \\ P^T & O \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{0} \end{bmatrix} \tag{3.7}$$

where now the matrices and arrays have the following dimensions: A is $N \times N$, P is $N \times M$, O is $M \times M$; **c, y** are $N \times 1$ and **d, 0** are $M \times 1$.

Remark. It seems ill-adapted for the use to formulate the general setting for the polynomial space $\mathbb{P}_{m-1}(\mathbb{R}^d)$ instead, as before, of $\mathbb{P}_m(\mathbb{R}^d)$. The reason will be explained later and, as we will see, it will be quite natural.

3.0.1 Conditionally positive definite matrices and functions

In order to prove that the augmented system (3.7) is non-singular we start with the easiest case of $m = 1$ (in any dimension d). This is equivalent to the reproduction of the polynomials of degree $m - 1 = 0$, i.e. the constants.

Definition 9. *A real symmetric matrix A, $N \times N$, is called* `conditionally positive semi-definite of order` $m = 1$ *if its associated quadratic form $\mathbf{c}^T A \boldsymbol{c} \ge 0$ for all $\boldsymbol{c} = (c_1, \ldots, c_N)^T \in \mathbb{R}^N$ that satisfy*

$$\sum_{i=1}^{N} c_i = 0. \tag{3.8}$$

If $\boldsymbol{c} \neq \mathbf{0}$ implies strictly inequality, i.e. $\mathbf{c}^T A \boldsymbol{c} > 0$, than A is called `conditionally positive of order one`*.*

Notice: the conditions (3.8) can be viewed as a condition of "ortogonality" w.r..t. constant functions.

The following theorem shows that the augmented system is uniquely solvable

Theorem 9. *Let A be real symmetric of order N that is conditionally postive definite (CPD) of order one. Let $P = (1, \dots, 1)^T$ be an $N \times 1$ array. Then the system*

$$\begin{bmatrix} A & P \\ P^T & O \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ d \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ 0 \end{bmatrix} \tag{3.9}$$

is uniquely solvable

Proof.. Assume that the solution $(\boldsymbol{c}, d)^T \in \mathbb{R}^{N+1}$ is a solution of the homogeneous system, i.e. with $\boldsymbol{y} = \boldsymbol{0}$. We then prove that $\boldsymbol{c} = \boldsymbol{0}$ and $d = 0$. From the first block, multiplying by $\boldsymbol{c}^T$, we have

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

an using the second equation $P^T \boldsymbol{c} = \boldsymbol{c}^T P = 0$ we get that $\boldsymbol{c}^T A \boldsymbol{c} = 0$. Since A is CPD of order 1, by asumption we get $\boldsymbol{c} = \boldsymbol{0}$. Moreover, from the first block, $A\boldsymbol{c} + dP = \boldsymbol{0}$ that implies $d = 0$. ■

Corollary 1. *For reproducing constant functions in $\mathbb{R}^d$, the system to be solved has the form (3.9).*

We are now ready to introduce `conditionally positive definite functions of order` m.

Definition 10. *A continuous function $\Phi : \mathbb{R}^d \to \mathbb{C}$ is said to be* `conditionally positive semi-definite of order` m *in $\mathbb{R}^d$, if*

$$\sum_{i=1}^{N} \sum_{j=1}^{N} c_i \bar{c}_j \Phi(\boldsymbol{x}_i - \boldsymbol{x}_j) \geq 0 \tag{3.10}$$

for any N set $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\} \subset \mathbb{R}^d$ of pairwise distinct points, and $\boldsymbol{c} = (c_1, \dots, c_N)^T \subset \mathbb{C}^N$ such that

$$\sum_{k=1}^{N} c_k p(\boldsymbol{x}_k) = 0$$

for any complex-valued polynomial p of degree $\leq m - 1$. The function Φ is then called `conditionally positive definite` *of order m on $\mathbb{R}^d$ if the quadratic form (3.11) vanishes only when $\boldsymbol{c} \equiv \boldsymbol{0}$.*

The first important fact concerning conditionally positive (semi)-definite functions is their `order`. To this aim holds the following important result.

Proposition 3. *A function which is conditionally positive (semi)-definite of order m is also conditionally positive (semi)-definite of any order $s \geq m$. Moreover, a function that is conditionally positive (semi)-definite of order m in $\mathbb{R}^d$ is also conditionally positive (semi)-definite of order m on $\mathbb{R}^k$ with $k \leq d$.*

Hence the natural question is to look for the `smallest possibile order` m. When speak of the order of such class of functions we will always refer to the minimal possible m.

Examples of CPD functions

Here we list the most important conditionally positive definite radial functions

- `Generalized multiquadrics`.

  $$\Phi(\boldsymbol{x}) = (1 + \|\boldsymbol{x}\|^2)^\beta, \;\; \mathbf{x} \in \mathbb{R}^d, \; \beta \in \mathbb{R} \backslash \mathbb{N}_0 \,,$$

  which are CPD of order $m = \lceil \beta \rceil$ (and higher).

  Notice that in the definition we have to exclude positive integer values for β, otherwise we are led to polynomials of even degree.

  Two cases: for $\beta = 1/2$ we obtain the well-known `Hardy multiquadric` that is CPD of order 1; for $\beta = 5/2$ we have a function which is CPD of order 3.

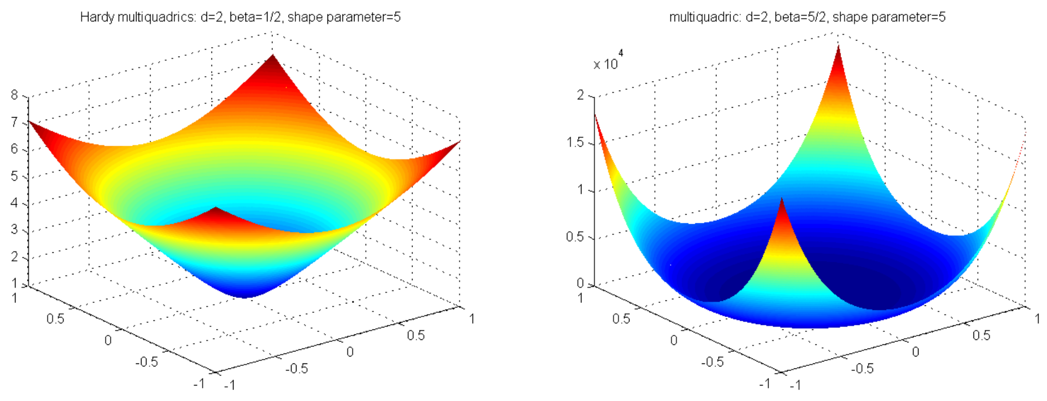


Figure 3.1: Multiquadrics. Left: Hardy multiquadric. Right: with $\beta = 5/2$.

- `Radial powers`

  $$\Phi(\mathbf{x}) = \|\mathbf{x}\|^\beta, \;\; \mathbf{x} \in \mathbb{R}^s, 0 < \beta \notin 2\mathbb{N} \,,$$

  which are CPD of order $m = \lceil \frac{\beta}{2} \rceil$ (and higher).

  For $\beta = 3$ we get a function which is CPD of order 2 and for $\beta = 5$ a function which is CPD of order 3. A property of these functions is that they are *shape parameter free*. This has the advantage that the user need not worry about finding a *good* (or the best) value for ϵ.

- `Thin-plate splines`

  $$\Phi(\mathbf{x}) = \|\mathbf{x}\|^{2\beta} \log \|\mathbf{x}\|, \;\; \mathbf{x} \in \mathbb{R}^s, \beta \in \mathbb{N} \,,$$

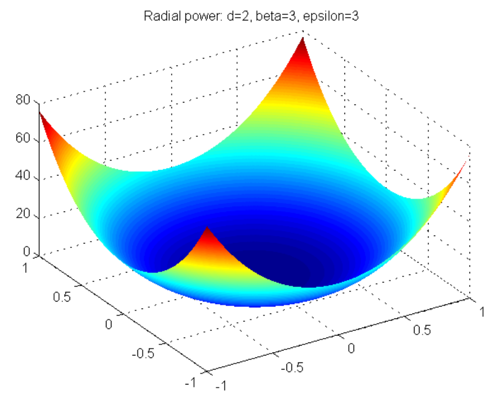


Figure 3.2: Radial powers. Left: with $\beta = 3$, $\epsilon = 3$. Right: with $\beta = 5$, $\epsilon = 3$.

which are CPD of order $m = \beta + 1$.

The classical TPS is for $\beta = 1$ that is CPD of order 2. For $\beta = 2$ we get a CPD function of order 3. Also TPS are *shape parameter free.*

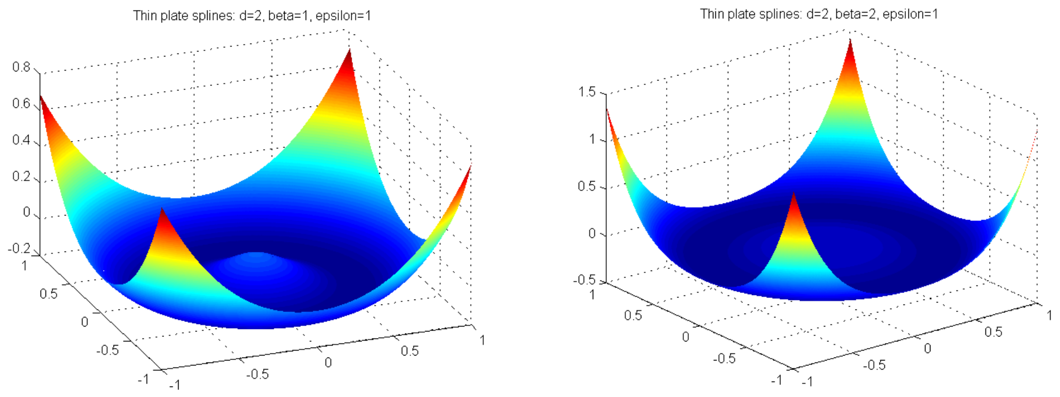


Figure 3.3: Thin plate splines. Left: for $\beta = 1$, $\epsilon = 1$. Right: for $\beta = 2$, $\epsilon = 1$.

◇◇

As for the PD case, the definition reduces to real coefficients and polynomials if the basis function is real-valued and even. This is the case when the function Φ is radial.

Theorem 10. *A continuous function* $\Phi : \mathbb{R}^d \to \mathbb{R}$ *is said to be* `conditionally positive definite of order` m *in* $\mathbb{R}^d$, *if*

$$\sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j \Phi(\boldsymbol{x}_i - \boldsymbol{x}_j) > 0 \tag{3.11}$$

for any N set $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\} \subset \mathbb{R}^d$ of pairwise distinct points, and $\boldsymbol{c} = (c_1, \dots, c_N)^T \subset \mathbb{R}^N \setminus \{0\}$ such that

$$\sum_{k=1}^{N} c_k p(\boldsymbol{x}_k) = 0$$

for any real-valued polynomial p of degree $\le m-1$.

The special case $m = 1$ appers already in the linear algebra literature and it is dealt with $\le$ and is then referred as *conditionally negative definite*. The constraints are simply $\sum_{i=1}^{N} c_i = 0$. Since the matrix A is CPD of order 1, it is PD in a subspace of dimension $N-1$ (or in general $N-M$ where $M = \dim(\mathbb{P}_{m-1}(\mathbb{R}^d))$. That is only $N-1$ (or $N-M$) eigenvalues are positive.

For conditionally positive definite functions of order $m = 1$ the following important property holds (which is Example 2 of Lecture 1).

Theorem 11. *Suppose Φ is CPD of order 1 and that $\Phi(0) \le 0$. Then the matrix $A \in \mathbb{R}^{N\times N}$, i.e. $A_{i,j} = \Phi(\boldsymbol{x}_i - \boldsymbol{x}_j)$, has one negative and $N-1$ positive eigenvalues. In particular it is invertible.*

Proof. From the Courant-Fischer theorem, we conclude that it has $N-1$ positive eigenvalues. Now, since $0 \ge N\Phi(0) = \mathtt{tr}(A) = \sum_{i=1}^{N} \lambda_i$, then A must have at least one negative eigenvalue. ■

For example, the generalized multiquadrics $\Phi(\boldsymbol{x}) = (-1)^{\lceil \beta \rceil}(1 + \|\boldsymbol{x}\|^2)^\beta,\ 0 < \beta < 1$ (which includes the Hardy's one, $\beta = 1/2$) satisfied the previous Theorem.

Finally we can prove a Theorem similar to Theorem 9.

Theorem 12. *Let A be real symmetric of order N that is conditionally positive definite (CPD) of order m on $\mathbb{R}^d$ and the points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$ form an $m-1$-unisolvent set. Then the system*

$$\begin{bmatrix} A & P \\ P^T & O \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{0} \end{bmatrix} \tag{3.12}$$

is uniquely solvable.

Proof.. Assume that the solution $(\boldsymbol{c}, \boldsymbol{d})^T \in \mathbb{R}^{N+M}$ is a solution of the homogeneous system, i.e. with $\boldsymbol{y} = \boldsymbol{0}$. We then prove that $\boldsymbol{c} = \boldsymbol{0}$ and $\boldsymbol{d} = \boldsymbol{0}$. From the first block, multiplying by $\boldsymbol{c}^T$, we have

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

an using the second equation $P^T \boldsymbol{c} = \boldsymbol{c}^T P = \boldsymbol{0}^T$ we get that $\boldsymbol{c}^T A \boldsymbol{c} = 0$. Since A is CPD of order m, by assumption we get $\boldsymbol{c} = \boldsymbol{0}$. The unisolvency of the data sites, i.e. P has columns linearly independent, and the fact that $\boldsymbol{c} = \mathbf{0}$ guarantee $\boldsymbol{d} = \mathbf{0}$ from the top block $A\boldsymbol{c} + P\boldsymbol{d} = \mathbf{0}$. ■

3.1 Exercises

1. Plot the most important *conditionally positive definite functions* in the square $[-1, 1]^2$.

   (a) `generalized multiquadrics`

   $$\Phi(\mathbf{x}) = (1 + \|\mathbf{x}\|^2)^\beta, \ \ \mathbf{x} \in \mathbb{R}^2, \ \beta \in \mathbb{R} \backslash \mathbb{N}_0$$

   in particular for $\beta = 1/2$ we have the Hardy multiquadric which is of (minimum) order 1 and that for $\beta = 5/2$ which is of (minimun) order 3.

   (b) `radial powers`

   $$\Phi(\mathbf{x}) = \|\mathbf{x}\|^\beta, \ \ \mathbf{x} \in \mathbb{R}^2, 0 < \beta \notin 2\mathbb{N}$$

   For example take $\beta = 3$ (which is CPD of order 2) and $\beta = 5$ (CPD of order 3). Verify furthermore that the power functions are *shape parameter free.*

   (c) `thin-plate splines`

   $$\Phi(\mathbf{x}) = \|\mathbf{x}\|^{2\beta} \log \|\mathbf{x}\|, \ \ \mathbf{x} \in \mathbb{R}^2, \beta \in \mathbb{N}$$

   The most important is the one for $\beta = 1$ which is CPD of order 2. For $\beta = 2$ we get a function which is CPD of order 3. Verify that also these functions are *shape parameter free.*

2. For a CPD function of order 1 (such as the Hardy multiquadric) check the Theorem 11.

Lecture 4

Error estimates

In evaluating the error between the interpolant P_f and the data values at some set $\Xi = \{\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_M\} \subset \mathbb{R}^d$ of *evaluation points* we can compute the `root-mean-square error`, that is

$$RMSE := \sqrt{\frac{1}{M}\sum_{j=1}^{M}(P_f(\boldsymbol{\xi}_j) - f(\boldsymbol{\xi}_j))^2} = \frac{1}{\sqrt{M}}\|P_f - f\|_2 \,. \tag{4.1}$$

The root-mean-square error (RMSE) is a frequently used measure of the differences between values predicted by a model or an estimator and the values actually observed. These individual differences are called *residuals* when the calculations are performed over the data sample that was used for estimation, and are called *prediction errors* when computed out-of-sample. The RMSE serves to aggregate the magnitudes of the errors in predictions for various times into a single measure of predictive power. RMSE is a good measure of accuracy, but only to compare forecasting errors of different models for a particular variable and not between variables, as it is scale-dependent. In practice, (4.1) is simply a quantitative error estimator.

Our goal is to provide error estimates for scattered data interpolation with (conditionally) positive definite functions. We start by considering the PD case.

4.1 Fill distance

The measure that is always used in approximation theory is the *fill-distance* or *mesh size*

$$h = h_{X,\Omega} := \sup_{\boldsymbol{x}\in\Omega} \min_{\boldsymbol{x}_i \in X} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2 \,, \tag{4.2}$$

which represents how well the data in X fill out the domain Ω and corresponds to the radius of the largest empty ball that can be placed among the data sites inside Ω.

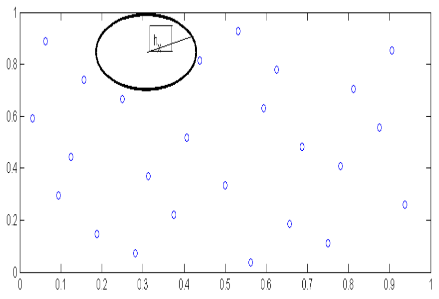


Figure 4.1: The fill distance of 25 Halton points $h \approx 0.2667$

In Matlab the fill distance can be determined by the line `h=max(min(DM_eval'))`, where `DM_eval` is the matrix consisting of the mutual distances between the evaluation points (for example a uniform grid in Ω) and the data set X. In Fig. 4.1 we show 25 Halton points and the corresponding fill distance computed on a grid of 11×11 evaluation points of $[0,1]^2$.

What we want to analysize, is whether the error $\|f - P_f^{(h)}\|_\infty \to 0$ as $h \to 0$, here $P_f^{(h)}$ indicates the interpolant depending on the fill-distance h. To understand the speed of convergence to zero, one has to understand the so-called *approximation order* of the interpolation process.

Definition 11. *We say that the process has* `approximation order` k *if*

$$\|f - P_f^{(h)}\|_p = \mathcal{O}(h^k), \quad k \to 0$$

here the norm is taken for $1 \le p \le \infty$*.*

For completeness, another quantity often used for stability analysis purposes (or finding "good" interpolation points) is the *separation distance* of the data sites X

$$q_X := \frac{1}{2} \min_{i \neq j} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$$

which represents the radius of the largest ball that can be placed around every point of X such there are no overlaps. In Matlab `qX=min(min(DM_data+eye(size(DM_data))))/2`, where `DM_data` is the matrix of distances among the data sites. We added the identity matrix in order to avoid that this minimum will be 0, which is the separation distance of every points from itself.

Finally, the ratio

$$\rho_{X,\Omega} := \frac{q_X}{h_{X,\Omega}}$$

known as *uniformity*, which can be identified as $\rho_{X,\Omega} = \lim_{Y \in X_\Omega} \rho_{X,\Omega}$ among all point sets $Y \in X_\Omega$ with X_Ω consisting, in some cases, of the Voronoi vertices used to decompose $\mathbb{R}^d$ with Voronoi tiles. Therefore if $\rho_{X,\Omega} \approx 1$ the data points are nearly equispaced in the Euclidean norm. It has been proved in [25] that, in contrast with polynomial interpolation, radial basis interpolants behaves better when the points are nearly equispaced,

4.2 Lagrange form of the interpolant

This idea goes back to Wu and Schaback [78] and consists in expressing the interpolant by means of *cardinal functions* as in the polynomial case.

Instead of solving the system $A\boldsymbol{c} = \boldsymbol{y}$ as we did in the previous lectures, we consider the (new) system

$$A\boldsymbol{u}^*(\boldsymbol{x}) = \boldsymbol{b}(\boldsymbol{x}) \tag{4.3}$$

where A is the $N \times N$ positive definite matrix (invertible!) $A_{i,j} = \Phi(\boldsymbol{x}_i - \boldsymbol{x}_j),\ i,j = 1,\ldots,N$, $\boldsymbol{u}^* = (u_1^*,\ldots,u_N^*)^T$, with the $u_j^*(\boldsymbol{x}_i) = \delta_{i,j}$ (i.e. cardinal functions), and $\boldsymbol{b} = (\Phi(\cdot - \boldsymbol{x}_1),\ldots,\Phi(\cdot - \boldsymbol{x}_N))^T$.

Why this is possible? Thanks to the following theorem

Theorem 13. *If Φ is a positive definite kernel on $\mathbb{R}^d$. Then, for any set of distinct points $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N$, there exist functions $u_j^* \in \mathrm{span}\{\Phi(\cdot - \boldsymbol{x}_j),\ j = 1,\ldots,N\}$ such that $u_j^*(\boldsymbol{x}_i) = \delta_{i,j}$.*

Here we enumerate a few facts in analogy with univariate fundamental Lagrange polynomials.

- The u_j^* can be determined as a ratio of determinants as for the fundamental Lagrange polynomials. Letting $V = \det(u_j^*(\boldsymbol{x}_i))$, and $V_{i,\boldsymbol{x}}$ the same determinant when $\boldsymbol{x}_i$ is substituted by a general point $\boldsymbol{x} \in \Omega$, then $u_i^*(\boldsymbol{x}) = V_{i,\boldsymbol{x}}/V$.
- The u_j^* do not depend on the data values (i.e. the f_j). In fact, once the data sites X and the kernel Φ are chosen, then they can be determined by solving the system (4.3). That is, they do depend on the data sites and the kernel.
- An important aspect, related to stability issues, is the choice of the data points. As proved in [25], the *quasi-uniform* points are always a good choice for RBF interpolation.

4.3 The power function

Another ingredient for understanding the error estimates is the *power function*. The starting point to define this function is the following quadratic form

$$Q(\boldsymbol{u}) = \Phi(\boldsymbol{0}) - 2\boldsymbol{u}^T\boldsymbol{b} + \boldsymbol{u}^T A\boldsymbol{u} \tag{4.4}$$

where the vector $\boldsymbol{b} \in \mathbb{R}^N$ is defined as in the previous section and $\boldsymbol{u}$ is any N dimensional vector.

Definition 12. *On $\mathbb{R}^d$ let us consider a subset Ω and a continuous kernel Φ which we assume PD. For any set $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\} \subset \Omega$ the* `power function` *is defined as follows*

$$P_{\Phi,X}(\boldsymbol{x}) = \sqrt{Q(\boldsymbol{u}^*(\boldsymbol{x}))}\,, \tag{4.5}$$

where $\boldsymbol{u}^*$ is the vector of the cardinal functions in Theorem 13.

Using formula (4.4), combined with the system that defines the cardinal functions (4.3), we get two alternatives ways to compute the power function

$$P_{\Phi,X}(\boldsymbol{x}) = \sqrt{\Phi(\boldsymbol{0}) - (\boldsymbol{u}^*(\boldsymbol{x}))^T \boldsymbol{b}(\boldsymbol{x})} = \sqrt{\Phi(\boldsymbol{0}) - (\boldsymbol{u}^*(\boldsymbol{x}))^T A \boldsymbol{u}^*(\boldsymbol{x})}\,, \quad (\texttt{first}) \tag{4.6}$$

$$P_{\Phi,X}(\boldsymbol{x}) = \sqrt{\Phi(\boldsymbol{0}) - (\boldsymbol{b}(\boldsymbol{x}))^T A^{-1} \boldsymbol{b}(\boldsymbol{x})}\,, \quad (\texttt{second}). \tag{4.7}$$

Notice that when Φ is a PD kernel then A is, therefore we get immediately the following bounds:

$$0 \le P_{\Phi,X}(\boldsymbol{x}) \le \sqrt{\Phi(\boldsymbol{0})}\,.$$

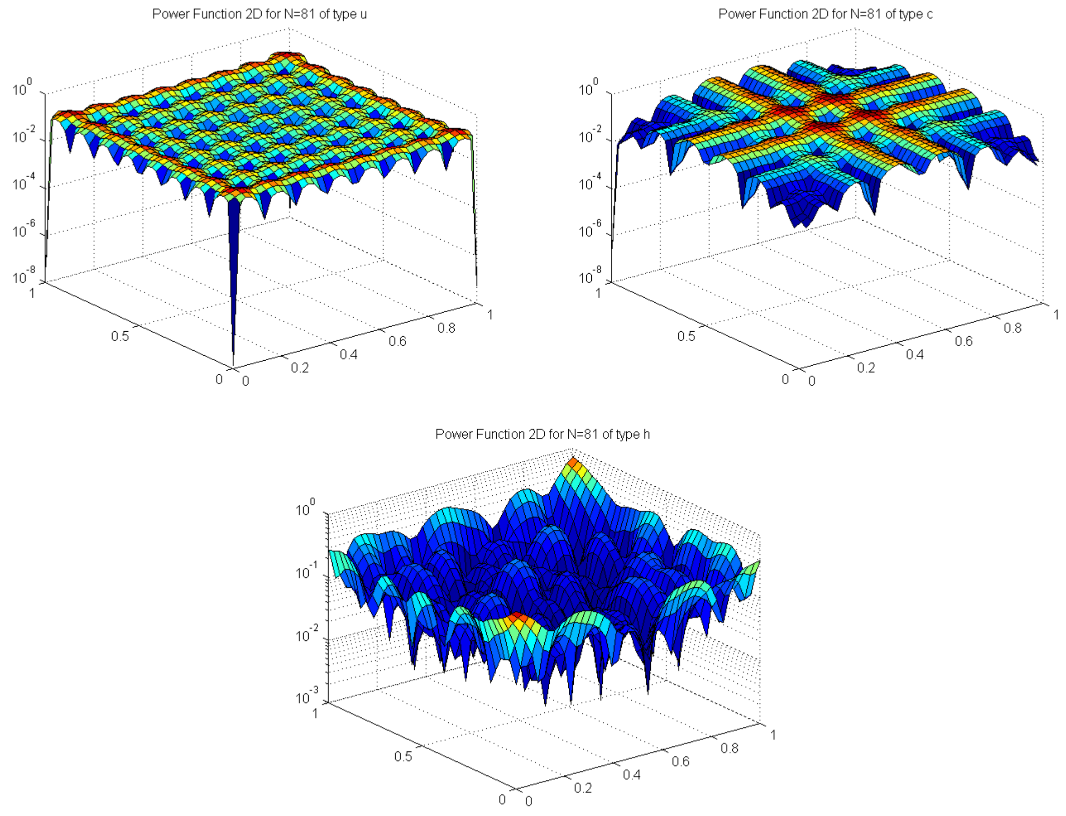


Figure 4.2: Power function for the Gaussian kernel with $\epsilon = 6$ on a grid of 81 uniform, Chebyshev and Halton points, respectively.

An interesting characterization of the power function is given in the next Theorem.

Theorem 14. *Let $\Omega \subset \mathbb{R}^d$ and Φ a PD kernel on $\mathbb{R}^d$. Let X as usual a set of N pairwise distinct points in Ω. The minimum of the quadratic form $Q(\boldsymbol{u})$ is when $\boldsymbol{u} = \boldsymbol{u}^*(\boldsymbol{x})$, that is*

$$Q(\boldsymbol{u}^*(\boldsymbol{x})) \le Q(\boldsymbol{u}), \text{ for all } \boldsymbol{u} \in \mathbb{R}^N .$$

Proof. Consider the formula (4.4), the minimum of this quadratic form is given by the solution of the linear system $A\boldsymbol{u} = \boldsymbol{b}(\boldsymbol{x})$ which, however yields the cardinal functions $\boldsymbol{u} = \boldsymbol{u}^*(\boldsymbol{x})$. $\blacksquare$

The power function, by definition, is a positive function, vanishing at the data sites, decreasing to zero as the number of data points increases. Therefore, if we take two data sets such that $Y \subset X$ then $P_{Y,\Omega} \ge P_{X,\Omega}$. This is referred as the *maximality property* of the power function.

As a final remark, the power function is defined similarly for conditionally positive definite functions.

4.4 Native space

The error bounds come rather naturally once we associate with each radial basic function a certain space of functions called `native space`. This space in connected to the so called *Reproducing Kernel Hilbert Space (RKHS)*. The theory of RKHS is beyond our aims, but for understanding a little better the error estimates that we will present, it is necessary to introduce some very basic notions of RKHS.

Definition 13. *A space of functions is called an* `Hilbert space` *if it is a real or complex inner product space that is also a complete metric space w.r.t. the distance induced by the inner product.*

Here the `inner product` between two functions f and g is thought as $(f,g) = \int_a^b f(x)g(x)dx$, in the real case or $(f,g) = \int_a^b f(x)\overline{g(x)}dx$, in the complex case, which has many of the familiar properties of the Euclidean (discrete) dot product.

Examples of Hilbert spaces are: any finite dimensional inner product space (for example $\mathbb{R}^n, \mathbb{C}^n$ equipped with the dot product of two vectors); the *Lebesgue spaces* L^p, *Sobolev spaces.* The space $\mathcal{C}([a,b])$ is an incomplete product space dense in $L^2([a,b])$ which is complete.

Definition 14. *Let H be a real Hilbert space of functions $f : \Omega \to \mathbb{R}$ with inner product $(\cdot,\cdot)_H$. A function $K : \Omega \times \Omega \to \mathbb{R}$ is called a RKHS for H if*

(i) $K(\cdot, \boldsymbol{x}) \in H$ *for all* $\boldsymbol{x} \in \Omega$

(ii) $f(\boldsymbol{x}) = (f, K(\cdot, \boldsymbol{x}))_H$ *for all* $f \in H$ *and all* $\boldsymbol{x} \in \Omega$.

The second is the *reproducing property.* In particular, if $f = K$ then we get the kernel K since $K(\boldsymbol{x}, \boldsymbol{y}) = (K(\cdot, \boldsymbol{y}), K(\cdot, \boldsymbol{x}))_H$ for all $\boldsymbol{x}, \boldsymbol{y} \in \Omega$.

Notice: the RKHS is known to be unique and that the kernel K is positive definite.

We now show that *"every positive definite radial basis function can be associated with a RKHS: its* `native space`*."*

From Definition 14, a space H should contains functions of the form

$$f(\cdot) = \sum_{i=1}^{N} c_i K(\cdot, \boldsymbol{x}_i)$$

provided $\boldsymbol{x}_j \in \Omega$. Moreover

$$\begin{aligned}
\|f\|_H^2 &= (f, f)_H = (\sum_{i=1}^{N} c_i K(\cdot, \boldsymbol{x}_i), \sum_{j=1}^{N} c_j K(\cdot, \boldsymbol{x}_j))_H \\
&= \sum_{i,j=1}^{N} c_i c_j (K(\cdot, \boldsymbol{x}_i), K(\cdot, \boldsymbol{x}_j))_H \\
&= \sum_{i,j=1}^{N} c_i c_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)\,.
\end{aligned}$$

Hence we define this (infinite dimensional) space

$$H_K(\Omega) := \text{span}\{K(\cdot, \boldsymbol{y}) : \ \boldsymbol{y} \in \Omega\} \tag{4.8}$$

with associated a *bilinear form*

$$(\sum_{i=1}^{N_K} c_i K(\cdot, \boldsymbol{x}_i), \sum_{j=1}^{N_K} d_j K(\cdot, \boldsymbol{y}_j))_K = \sum_{i,j=1}^{N_K} c_i d_j K(\boldsymbol{x}_i, \boldsymbol{y}_j)$$

where $N_k = \infty$ is also possible.

The last observation is that, this bilinear form defines an inner product on $H_K(\Omega)$, so that $H_k(\Omega)$ is a *pre-Hilbert space*, that means that it is not complete.

The *native space* for the kernel K, indicated as $\mathcal{N}_K(\Omega)$ (or if not confusion arises, simply $\mathcal{N}$), is the completition of $H_K(\Omega)$ w.r.t. the K-norm $\|\cdot\|_K$, so that $\|f\|_K = \|f\|_{\mathcal{N}}$ for all $f \in H_K(\Omega)$. For details, please refer to the book by Wendland [74].

Remark. In dealing with positive definite (translation invariat) functions Φ, we will write

$$\Phi(\boldsymbol{x}-\boldsymbol{y}) = K(\boldsymbol{x},\boldsymbol{y})\,,$$

so that K is taken in $\Omega \times \Omega$ instead of simply Ω.

4.5 Generic error estimates

We quote here two results that gives a flavour of the topic. The first theorem gives a pointwise estimate based on the power function, whose proof can be found in [32, p. 117-118] while the second uses the fill-distance (the proof is again in [32, p. 121-122]).

Theorem 15. *Let $\Omega \subset \mathbb{R}^d$ and $\Phi \in \mathcal{C}(\Omega\times\Omega)$ be PD on $\mathbb{R}^d$. Let $X = \{\boldsymbol{x}_1, \ldots, , \boldsymbol{x}_n\}$ be a set of distinct points. Take a function $f \in \mathcal{N}_\Phi(\Omega)$ and denote with P_f its interpolant on X. Then, for every $\boldsymbol{x} \in \Omega$*

$$|f(\boldsymbol{x}) - P_f(\boldsymbol{x})| \le P_{\Phi,X}(\boldsymbol{x})\|f\|_{\mathcal{N}_\Phi(\Omega)}\,. \tag{4.9}$$

where the norm of f is the native space norm.

We can express such error by means of the *fill distance*.

Theorem 16. *Let $\Omega \subset \mathbb{R}^d$ and $\Phi \in \mathcal{C}^{2k}(\Omega\times\Omega)$ be symmetric and positive definite. Let $X = \{\boldsymbol{x}_1, \ldots, , \boldsymbol{x}_n\}$ be a set of distinct points. Take a function $f \in \mathcal{N}_\Phi(\Omega)$ and its interpolant P_f on X. Then, there exist positive constants h_0 and C (independent of $\boldsymbol{x}$, f and Φ), with $h_{X,\Omega} \le h_0$, such that*

$$|f(\boldsymbol{x}) - P_f(\boldsymbol{x})| \le C h_{X,\Omega}^k \sqrt{C_\Phi(\boldsymbol{x})}\|f\|_{\mathcal{N}_\Phi(\Omega)}\,. \tag{4.10}$$

and

$$C_\Phi(\boldsymbol{x}) = \max_{|\beta|=2k} \max_{\boldsymbol{w},\boldsymbol{z}\in\Omega\cup B(\boldsymbol{x},c_2 h_{X,\Omega})} |D_2^\beta \Phi(\boldsymbol{w},\boldsymbol{z})|\,.$$

Comparing (4.9) and (4.10) we get a bound of the power function in terms of the fill-distance

$$P_{\Phi,X}(\boldsymbol{x}) \le C h_{X,\Omega}^k \sqrt{C_\Phi(\boldsymbol{x})}\,.$$

Moreover the Theorem 16 says that interpolation with a $\mathcal{C}^{2k}$ kernel Φ has approximation order k. This means that for kernels infinitely smooth, such as Gaussians, Laguerre-Gaussians, Poisson and generalized inverse multiquadrics, the bound above is arbitrarely high. On the contrary, Matérn, Whittaker radial functions have approximation order limited by the smoothness of the basic function ϕ.

One more observation is that the above estimates consider $f \in \mathcal{N}_{\Phi(\Omega)}$. There exist similar estimates for $f \notin \mathcal{N}_{\Phi(\Omega)}$ (see e.g. [32, §15.3]).

4.6 Strategies in reducing the interpolation error

This last section aims to give some insights to the problem of the choice of shape parameter ϵ in order to get the smallest (possible) interpolation error. In the recent literature there have been exploited various strategies. Here, we present only three of them, which turn out to be indeed the most used by practictioners.

In all that follows, we assume to use the same kernel Φ, we use *only one value* ϵ to scale all basis functions uniformly. The number of data points could be changed by comparison of results.

4.6.1 Trial and Error

It is the simplest approach. It consists in performing various interpolation experiments with different values of the shape parameter. The "best" parameter, say ϵ^* will be the one that minimize the interpolation error. In Figure 4.3 we plot the interpolation max-error varying ϵ for different data points, using the Gaussian kernel in the univariate case. The minimum of every curve gives the "optimal" value.

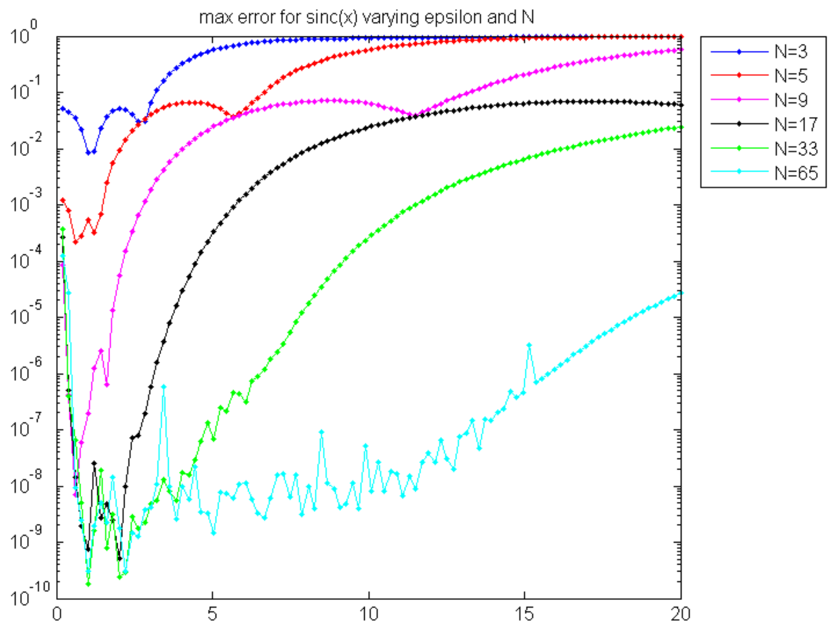


Figure 4.3: Trial and error strategy for the interpolation of the 1-dimensional sinc function by the Gaussian for $\epsilon \in [0, 20]$, taking 100 values of ϵ and for different equispaced data points.

4.6.2 Power function

This is simply connected to the error analysis presented in the previous section (cf. formula (4.9)). Once we have decided which Φ and data set X to use, we calculate the power function on scaled version of the kernel in order to optimize the error component that is independent of f. This approach is similar to the Trial and Error strategy and has the limit to forget the second part of the error, i.e. the one that depends on the basis function via the native space norm of f. In Figure 4.5 we plot the sup-norm of the power function for 500 values of $\epsilon \in [0, 20]$ for the Gaussian kernel and the set of uniform data points X with $N = 9, 25, 81, 289$. This strategy is implemented in the M-file `Powerfunction2D.m` in the Matlab codes provided in [32].

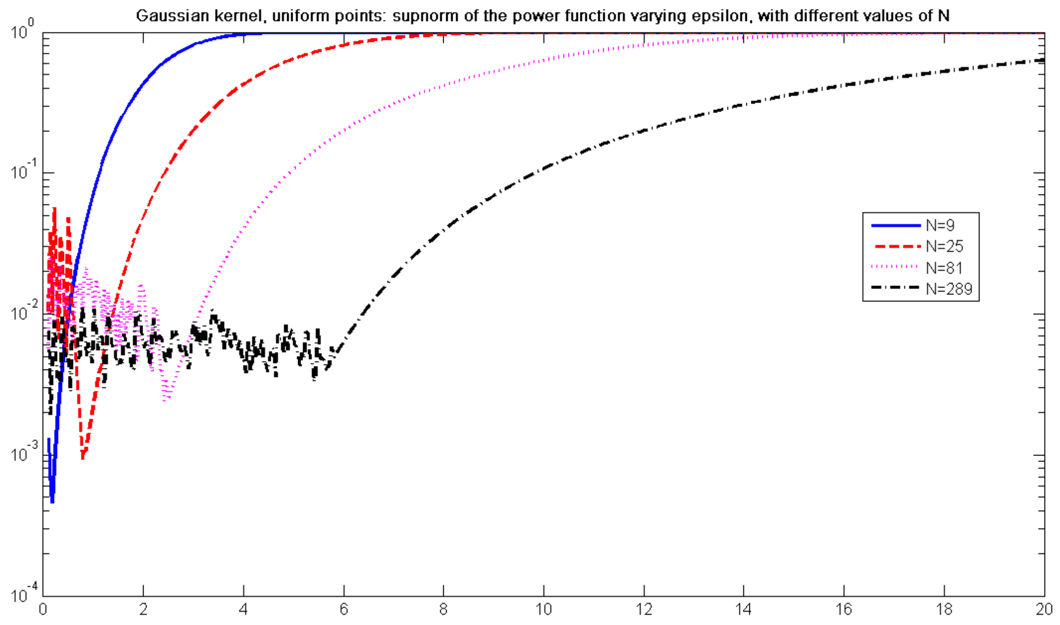


Figure 4.4: The sup-norm of the 2D power function on uniform points for the Gaussian kernel, varying $\epsilon \in [0, 20]$ for different values of N

4.6.3 Cross validation

This method is popular in the statistics literature, known in the case the 2-norm is used as *PRESS* (Predictive REsidual Sum of Squares). The "optimal" value ϵ^* is obtained minimizing the (least-squares) error for a fit of the data values based on an interpolant for which one of the centers is *left out*. In details, we start by constructing $P_{f,k}$, the radial basis function interpolant to the data $\{f_1, \ldots, f_{k-1}, f_{k+1}, \ldots, f_N\}$, that is

$$P_{f,k}(\boldsymbol{x}) = \sum_{i=1, i\neq k}^{N} c_{i,k}\Phi(\boldsymbol{x} - \boldsymbol{x}_i)$$

such that

$$P_{f,k}(\boldsymbol{x}_i) = f_i, \quad i = 1, \ldots, k-1, k+1, \ldots, N\,.$$

Then we compute the error at the point $\boldsymbol{x}_k$, the one not used by $P_{f,k}$

$$E_k = f_k - P_{f,k}(\boldsymbol{x}_k)\,.$$

Then the "quality" of the fit is determined by the (sup) norm of the vector $E = [E_1, \ldots, E_N]^T$.

In the experiments, people add a loop on ϵ in order to compare the error norms for different values of the shape parameter, choosing for ϵ^* the one that yields the minimum error norm. This method in general is quite expensive from the computation point of view (it has a complexity of $\mathcal{O}(N^4)$ flops). There is a way to "accelerate" the method, by computing E_k as

$$E_k = \frac{c_k}{A^{-1}_{k,k}}$$

where c_k is the k-th coefficient in the interpolant P_f based on *all* the data points and $A^{-1}_{k,k}$ is the k-th diagonal element of the inverse of the corresponding collocation matrix. Since both c_k and A^{-1} will be computed once for each value of ϵ, this results in $\mathcal{O}(N^3)$ flops. This strategy is implemented in the M-file `LOOCV.m` in the Matlab codes provided in [32].

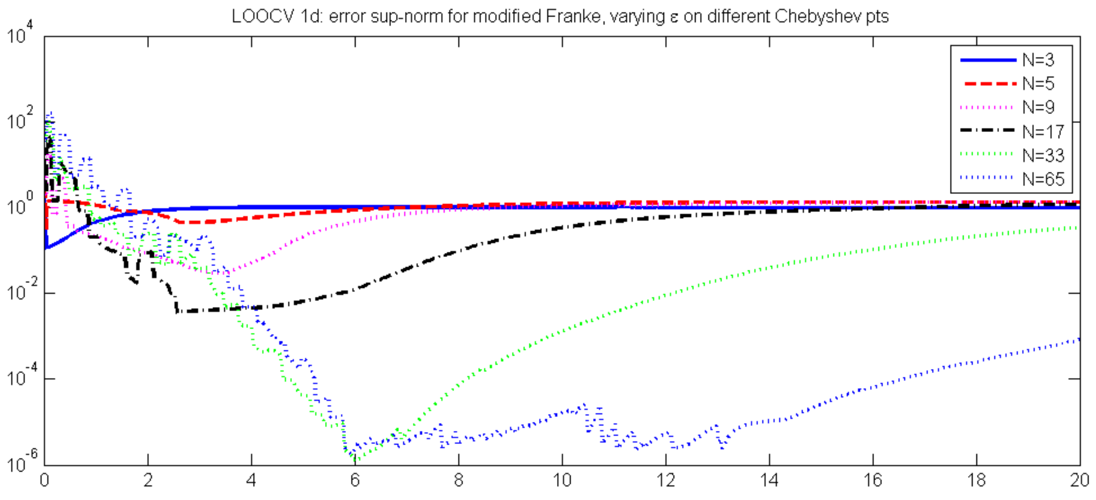


Figure 4.5: LOOCV 1d: sup norm of the error on Chebyshev points for the Gaussian kernel, varying $\epsilon \in [0, 20]$ for different values of N

4.7 Exercises

1. Find the *optimal shape parameter*, ϵ_{opt}, by means of the *trial & error* strategy for the following univariate functions:

   (a)
   $$f_1(x) = \texttt{sinc}(x) = \frac{\sin \pi x}{\pi x}\,.$$

   (b) *variant of the Franke function*
   $$f_2(x) = \frac{3}{4}\left(\mathrm{e}^{-(9x-2)^2/4} + \mathrm{e}^{-(9x+1)^2/49}\right) + \frac{1}{2}\mathrm{e}^{-(9x-7)^2/4} - \frac{1}{10}\mathrm{e}^{-(9x-4)^2}\,,$$

   For each of the f_i, $i = 1, 2$ produce a table of the form

N	$\|P_{f_i} - f_i\|_\infty$	ϵ_{opt}
3		
5		
9		
17		
33		
65		

   where, for each N, ϵ_{opt} corresponds to the minimum of the error curve in the sup-norm, computed by varying the shape parameter $\epsilon \in [0, 20]$. As radial basis function for the P_{f_i} take the Gaussian.

2. Plot the *power function* in 2-dimension for the *Gaussian kernel* with $\epsilon = 6$ on a grid of $N = 9^2 = 81$ equispaced, Chebyshev and Halton points in $[-1, 1]^2$. We'll see that the power function will depend on the chosen points.

   Verify that $P_{X,\Omega}(\boldsymbol{x}_i) = 0$ for all $\boldsymbol{x}_i \in X$. Show how varies the maximum value of the power function as N increases.

   Use the M-file `Powerfunction2D.m`.

3. Plot $\|P_{\Phi,X}\|_\infty$ for the Gaussian kernel in 2-dimensions, by using for the power function the formula
   $$P_{\Phi,X}(\boldsymbol{x}) = \sqrt{\Phi(\boldsymbol{0}) - (\boldsymbol{b}(\boldsymbol{x}))^T A^{-1}\, \boldsymbol{b}(\boldsymbol{x})}$$
   with A representing the interpolation matrix and $\boldsymbol{b}(\boldsymbol{x}) = [\Phi(\boldsymbol{x}-\boldsymbol{x}_1), \cdots, \Phi(\boldsymbol{x}-\boldsymbol{x}_N)]^T$, by varying $\epsilon \in [0, 20]$, and $N = 9,\ 25,\ 81,\ 289$. Take equispaced points both as centers and evaluation points of the power function.

   Make a table similar to the one of the previous exercise adding one column for the condition number of A corresponding to the "optimal" shape parameter. Use the function `Powerfunction2D.m` for computing the 2-dimensional power function.

Lecture 5

The Partition of Unity method

Local techniques, such as the PU scheme, have been used for interpolation since around 1960 [51, 69]. Later, they have been coupled with RBFs [32, 75]. Moreover, the PU method for solving Partial Differential Equations (PDEs), first introduced in the mid 1990s in [7, 56], is nowadays a popular technique [64, 68].

5.1 The local method

The basic idea of the PU method is to start with a partition of the open and bounded domain Ω into M *subdomains or patches* Ω_j, such that $\Omega \subseteq \cup_{j=1}^{M} \Omega_j$, with some mild overlap among them. In other words, the subdomains must form a covering of the domain and moreover, the overlap must be sufficient so that each interior point $\boldsymbol{x} \in \Omega$ is located in the interior of at least one patch Ω_j. The overlap condition is illustrated in Figure 5.1. Specifically, a 2D view of a PU structure, covering a set of scattered data in the unit square and satisfying the above mentioned properties, is shown in the left frame. In this particular example we consider circular patches, but other shapes can be considered. In the right frame we plot a PU structure which does not satisfy the overlap condition.

Furthermore, according to [75], some additional assumptions on the regularity of the covering $\{\Omega_j\}_{j=1}^{M}$ are required.

Definition 15. *$\Omega \subseteq \mathbb{R}^d$ satisfies an interior cone condition if there exist an angle $\theta \in (0, \pi/2)$ and a radius $\gamma > 0$ such that, for all $\boldsymbol{x} \in \Omega$, a unit vector $\boldsymbol{\xi}(\boldsymbol{x})$ exists such that the cone*

$$C = \{\boldsymbol{x} + \lambda \boldsymbol{y} : \boldsymbol{y} \in \mathbb{R}^d, ||\boldsymbol{y}||_2 = 1, \boldsymbol{y}^T \boldsymbol{\xi}(\boldsymbol{x}) \geq cos(\theta), \lambda \in [0, \gamma]\},$$

is contained in Ω.

Definition 16. *Suppose that $\Omega \subseteq \mathbb{R}^d$ is bounded and $\mathcal{X} = \{\boldsymbol{x}_i, i = 1, \ldots, N\} \subseteq \Omega$ is given. An open and bounded covering $\{\Omega_j\}_{j=1}^{M}$ is called regular for $(\Omega, \mathcal{X})$ if the following properties are satisfied:*

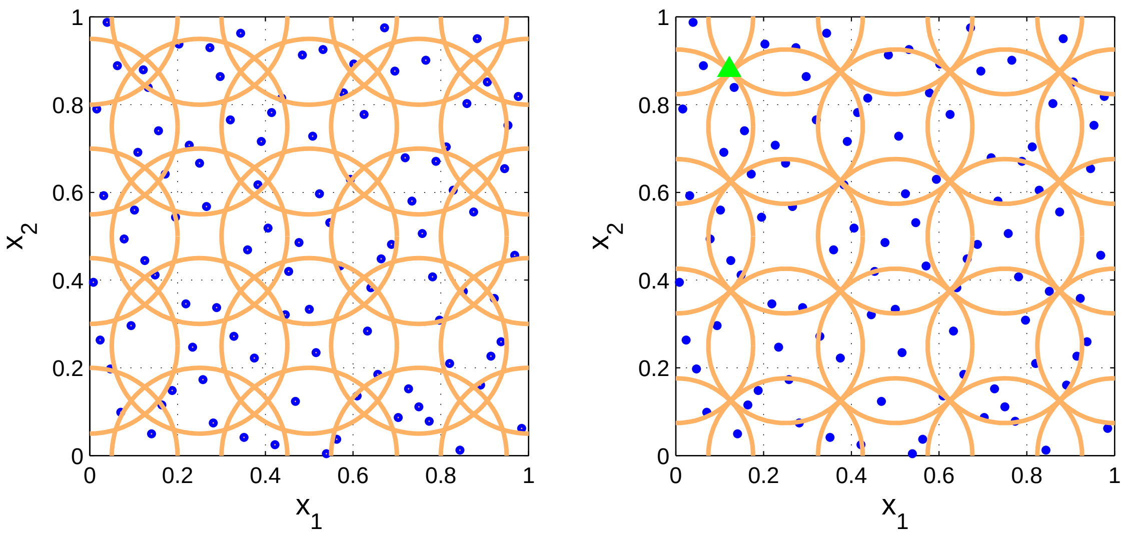


Figure 5.1: An illustrative example of PU subdomains covering the domain $\Omega = [0,1]^2$ and satisfying the overlap condition (left). In the right frame the covering does not satisfy the overlap condition; a critical point where no overlap occurs is marked with a green triangle. The blue dots represent a set of scattered data and the orange circles identify the PU subdomains.

i. *for each $\boldsymbol{x} \in \Omega$, the number of subdomains Ω_j, with $\boldsymbol{x} \in \Omega_j$, is bounded by a global constant C,*

ii. *each subdomain Ω_j satisfies an interior cone condition,*

iii. *the local fill distances $h_{\mathcal{X}_{N_j}}$ are uniformly bounded by the global fill distance $h_{\mathcal{X}}$, where $\mathcal{X}_{N_j} = \mathcal{X} \cap \Omega_j$.*

Associated to these subdomains, a partition of unity, i.e. a family of compactly supported, non-negative, continuous functions W_j, with $\operatorname{supp}(W_j) \subseteq \Omega_j$ and such that

$$\sum_{j=1}^{M} W_j(\boldsymbol{x}) = 1, \quad \boldsymbol{x} \in \Omega,$$

is considered. In addition, according to [75], we suppose that $\{W_j\}_{j=1}^{M}$ is a k-stable partition of unity, i.e. for every $\boldsymbol{\beta} \in \mathbb{N}^d$, with $|\boldsymbol{\beta}| \le k$, there exists a constant $C_{\boldsymbol{\beta}} > 0$ such that

$$\left\| D^{\boldsymbol{\beta}} W_j \right\|_{L^\infty(\Omega_j)} \le \frac{C_{\boldsymbol{\beta}}}{\left(\sup_{\boldsymbol{x},\boldsymbol{y} \in \Omega_j} \|\boldsymbol{x} - \boldsymbol{y}\|_2 \right)^{|\boldsymbol{\beta}|}}, \quad j = 1, \ldots, M.$$

Among several weight functions, a possible choice is given by the Shepard's weights [69], i.e.

$$W_j(\boldsymbol{x}) = \frac{\bar{W}_j(\boldsymbol{x})}{\displaystyle\sum_{k=1}^{M} \bar{W}_k(\boldsymbol{x})}, \quad j = 1, \ldots, M,$$

where $\bar{W}_j$ are compactly supported functions, with support on Ω_j. Moreover, such family $\{W_j\}_{j=1}^M$ forms a partition of unity.

Once we choose the partition of unity $\{W_j\}_{j=1}^M$, the global interpolant is formed by the weighted sum of M local approximants P_{f_j}, i.e.

$$\mathcal{I}(\boldsymbol{x}) = \sum_{j=1}^{M} P_{f_j}(\boldsymbol{x}) W_j(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega. \tag{5.1}$$

Remark. Note that the assumptions in Definition 16 lead to the requirement that the number of subdomains is proportional to the number of data. Furthermore, the first property ensures that the sum in (5.1) is actually a sum over at most C summands. The fact that C is independent of N, unlike M, is essential to avoid loss of convergence orders. Moreover, for an efficient evaluation of the global interpolant it is crucial that only a constant number of local approximants has to be evaluated. It means that it should be possible to locate those C indices in constant time.

In particular here P_{f_j} denotes a RBF interpolant defined on a subdomain Ω_j. For instance, supposing to have a strictly positive definite function, the j-th local interpolant assumes the form

$$P_{f_j}(\boldsymbol{x}) = \sum_{k=1}^{N_j} c_k^j \phi(||\boldsymbol{x} - \boldsymbol{x}_k^j||_2), \tag{5.2}$$

where N_j indicates the number of points on Ω_j and $\boldsymbol{x}_k^j \in \mathcal{X}_{N_j}$, with $k = 1, \ldots, N_j$. The coefficients $\{c_k^j\}_{k=1}^{N_j}$ in (5.2) are determined by enforcing the N_j local interpolation conditions

$$P_{f_j}(\boldsymbol{x}_i^j) = f_i^j, \quad i = 1, \ldots, N_j.$$

Thus, in case of strictly positive definite functions, the problem of finding the PU interpolant (5.1) reduces to solving M linear systems of the form

$$A_j \boldsymbol{c}_j = \boldsymbol{f}_j,$$

where $\boldsymbol{c}_j = (c_1^j, \ldots, c_{N_j}^j)^T$, $\boldsymbol{f}_j = (f_1^j, \ldots, f_{N_j}^j)^T$ and $A_j \in \mathbb{R}^{N_j \times N_j}$ is

$$A_j = \begin{pmatrix} \phi(||\boldsymbol{x}_1^j - \boldsymbol{x}_1^j||_2) & \cdots & \phi(||\boldsymbol{x}_1^j - \boldsymbol{x}_{N_j}^j||_2) \\ \vdots & \ddots & \vdots \\ \phi(||\boldsymbol{x}_{N_j}^j - \boldsymbol{x}_1^j||_2) & \cdots & \phi(||\boldsymbol{x}_{N_j}^j - \boldsymbol{x}_{N_j}^j||_2) \end{pmatrix}.$$

Moreover, since the functions W_j, $j = 1, \dots, M$, form a partition of unity, if the local fits P_{f_j}, $j = 1, \dots, M$, satisfy the interpolation conditions then the global PU approximant inherits the interpolation property [32, 75].

5.1.1 Error bounds for radial basis function partition of unity interpolants

In order to formulate error bounds, we need to define the space $C_\nu^k(\mathbb{R}^d)$ of all functions $f \in C^k$ whose derivatives of order $|\boldsymbol{\beta}| = k$ satisfy $D^{\boldsymbol{\beta}} f(\boldsymbol{x}) = \mathcal{O}(||\boldsymbol{x}||_2^\nu)$ for $||\boldsymbol{x}||_2 \longrightarrow 0$. We are now able to give the following convergence result [75].

Theorem 17. *Let $\Omega \subseteq \mathbb{R}^d$ be open and bounded and suppose that $\mathcal{X} = \{\boldsymbol{x}_i, i = 1, \dots, N\} \subseteq \Omega$. Let $\phi \in C_\nu^k(\mathbb{R}^d)$ be a strictly conditionally positive definite function. Let $\{\Omega_j\}_{j=1}^M$ be a regular covering for $(\Omega, \mathcal{X})$ and let $\{W_j\}_{j=1}^M$ be k-stable for $\{\Omega_j\}_{j=1}^M$. Then the error between $f \in \mathcal{N}_\phi(\Omega)$, where $\mathcal{N}_\phi$ is the native space of ϕ, and its PU interpolant* (5.1)*, with $P_{f_j} \in span\{\Phi(\cdot, \boldsymbol{x}), \boldsymbol{x} \in \mathcal{X}_N \cap \Omega_j\}$, can be bounded by*

$$|D^{\boldsymbol{\beta}} f(\boldsymbol{x}) - D^{\boldsymbol{\beta}} \mathcal{I}(\boldsymbol{x})| \le C' h_{\mathcal{X}}^{(k+\nu)/2-|\boldsymbol{\beta}|} ||f||_{\mathcal{N}_\phi(\Omega)},$$

for all $\boldsymbol{x} \in \Omega$ and all $|\boldsymbol{\beta}| \le k/2$.

Remark. If we compare the result reported in Theorem 17 with the global error estimates shown in the previous sections [74], we can see that the PU interpolant preserves the local approximation order for the global fit. Hence, we can efficiently compute large RBF interpolants by solving small interpolation problems and then combine them together with the global partition of unity $\{W_j\}_{j=1}^M$.

5.1.2 Partitioning data structures

In the PU setting, an important issue is the one of organizing points among the subdomains. In literature, techniques as *kd-trees*, which allow to partition data in a k-dimensional space, and related searching procedures have already been designed [2, 20]. Even if such techniques work with high dimensions, they are not specifically implemented for the PU method.

Here, we present a versatile scheme for multivariate interpolation which makes use of the so-called Integer-based Partitioning Structure (I-PS) and a related searching procedure [15]. It strictly depends on the size of the PU subdomains and allows to deal with a truly large number of data with a relatively low computational complexity. Such procedure follows from the results shown in [1, 12, 13, 14], where efficient searching procedures based on the partition of the underlying domains in strips or crossed strips are considered.

More precisely, the I-PS for bivariate and trivariate interpolation consists in covering, at first, the domain with several non-overlapping small squares or cubes, named *blocks*. Then, the usually large scattered data set is distributed among the different blocks rounding off to an integer value. We point out that the MATLAB software is made available to the scientific community in a downloadable free package `http://hdl.handle.net/2318/1527447`.

In what follows, we fix for simplicity $\Omega = [0,1]^d$. Assuming to have a nearly uniform node distribution, M is a suitable number of PU subdomains on Ω if [11, 32]

$$\frac{N}{M} \approx 2^d.$$

Here we take

$$M = \left\lfloor \frac{N^{1/d}}{2} \right\rfloor^d.$$

Also the PU radius δ must be carefully chosen. In fact, the patches must be a covering of the domain Ω and must satisfy the overlap condition. For instance, the required property can be fulfilled taking the radius

$$\delta = \frac{r}{M^{1/d}},$$

with $r \in \mathbb{R}$, $r \geq 1$.

The partitioning procedure presented in what follows is a partitioning data scheme based on storing points into different blocks, which are obtained from the subdivision of the data set into several squares or cubes.

The number q of blocks along one side of Ω is strictly linked to the PU radius δ and is given by

$$q = \left\lceil \frac{1}{\delta} \right\rceil. \tag{5.3}$$

From (5.3) we can deduce that the I-PS depends on the construction of the PU subdomains. In such framework we will be able to get an efficient procedure to find the nearest points.

Thus, after defining the width of the blocks as in (5.3), we number blocks from 1 to q^M. In a 2D context they are numbered from bottom to top, left to right, see Figure 5.2. For trivariate data sets, starting from the order shown in Figure 5.2, we continue numbering blocks along the quote as well.

Furthermore, we need to perform several procedures enabling us to answer the following queries, respectively known as *containing query* and *range search*:

i. given a PU centre $\bar{\boldsymbol{x}}_j \in \Omega$, return the k-th block containing $\bar{\boldsymbol{x}}_j$,

ii. given a set of data points $\mathcal{X}$ and a subdomain Ω_j, find all points located in that patch.

Given a PU centre $\bar{\boldsymbol{x}}_j$, if k_m is the index of the strip parallel to the subspace of dimension $d-1$ generated by x_p, $p = 1, \ldots, d$, and $p \neq m$, containing the m-th coordinate of $\bar{\boldsymbol{x}}_j$, then the index of the k-th block containing the PU centre is

$$k = \sum_{m=1}^{d-1} (k_m - 1)\, q^{d-m} + k_d. \tag{5.4}$$

As example in a 2D framework, the PU centre plotted in Figure 5.2 belongs to the k-th block, with $k = 4q + 3$; in fact here $k_1 = 5$ and $k_2 = 3$.

To find the indices k_m, $m = 1, \ldots, d$, in (5.4), we use an integer-based procedure consisting in rounding off to an integer value. Specifically, for each PU centre $\bar{\boldsymbol{x}}_j = (\bar{x}_{j1}, \ldots, \bar{x}_{jd})$, we have that

$$k_m = \left\lceil \frac{\bar{x}_{jm}}{\delta} \right\rceil.$$

Exactly the same procedure is adopted in order to store both the scattered data and the evaluation points into the different blocks.

Specifically, supposing that the j-th PU centre belongs to the k-th block, the integer-based searching procedure searches for all data lying in the j-th subdomain among those lying on the k-th neighborhood, i.e. in the k-th block and in its $3^d - 1$ neighboring blocks, see Figure 5.2. In particular, the partitioning structure based on blocks enables us to examine in the searching process at most $3^d - 1$ blocks. In fact, when a block lies on the boundary of the bounding box, we reduce the number of neighboring blocks to be considered.

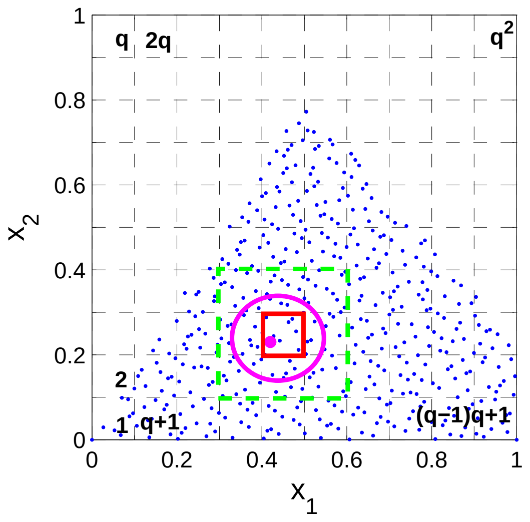


Figure 5.2: An example of a 2D partitioning structure: the k-th block (red solid line), a subdomain centre belonging to the k-th block (magenta circle) and the neighborhood set (green dashed line).

5.1.3 Complexity analysis

The I-PS, after organizing the scattered data into the different blocks, given a subdomain Ω_j searches for all the points lying on Ω_j in a reduced number of blocks. Specifically, in order to store the scattered data among the different blocks, it makes use of an integer-based procedure that assigns to each node $\boldsymbol{x}_i$, $i = 1, \ldots, N$, the corresponding block. This step requires $\mathcal{O}(N)$ time. Then, we apply the optimized searching routine. Supposing to have quasi-uniform data, it runs in a constant time.

In Table 5.1, we sum up the the total computational cost of the integer-based partitioning and searching procedures, compared with kd-trees.

integer-based structure	kd-tree structure	integer-based search	kd-tree search
$\mathcal{O}(N)$	$\mathcal{O}(dN \log N)$	$\mathcal{O}(1)$	$\mathcal{O}(\log N)$

Table 5.1: Computational costs concerning integer-based and the kd-tree routines.

Since from Definition 16, the number of centres in each PU subdomain is bounded by a constant, we need $\mathcal{O}(1)$ space and time for each patch to solve the local RBF interpolation problems. In fact, to get the local interpolants, we have to solve M linear systems of size $N_j \times N_j$, with $N_j \ll N$, thus requiring a running time $\mathcal{O}(N_j^3)$, $j = 1, \ldots, M$, for each patch. Besides reporting the points in each patch in $\mathcal{O}(1)$, as the number M of PU subdomains is bounded by $\mathcal{O}(N)$, this leads to $\mathcal{O}(N)$ space and time for solving all of them. Finally, we have to add up a constant number of local RBF interpolants to get the value of the global fit (5.1). This can be computed in $\mathcal{O}(1)$ time.

5.2 Modeling 3D objects via partition of unity interpolation

A common problem in computer aided design and computer graphics is the reconstruction of surfaces defined in terms of point cloud data, i.e. a set of unorganized points in 3D. Such applications arise in computer graphics, modeling complicated 3D objects or in medical imaging (see e.g. [16, 19, 60, 79]).

5.2.1 The implicit approach

An approach to obtain a surface that fits the given 3D point cloud data is based on the use of implicit surfaces defined in terms of some meshfree approximation methods, such as RBF interpolant [32]. Further details can also be found in [9, 10, 61, 73, 77].

Given a point cloud data set $\mathcal{X} = \{\boldsymbol{x}_i \in \mathbb{R}^3,\ i = 1, \ldots, N\}$, belonging to an unknown two dimensional manifold $\mathcal{M}$, namely a surface in $\mathbb{R}^3$, we seek another surface $\mathcal{M}^*$ that approximates $\mathcal{M}$.

For the implicit approach, we think of $\mathcal{M}$ as the surface of all points $\boldsymbol{x} \in \mathbb{R}^3$ satisfying the implicit equation

$$f(\boldsymbol{x}) = 0, \tag{5.5}$$

for some function f. So it implicitly defines the surface $\mathcal{M}^*$. This means that the equation (5.5) is the zero iso-surface of the trivariate function f, and therefore this iso-surface coincides with $\mathcal{M}$. The surface $\mathcal{M}$ can be constructed via PU interpolation. Unfortunately, the solution of this problem, by imposing the interpolation conditions (5.5), leads to the trivial solution, given by the identically zero function [19].

The key to finding the interpolant of the trivariate function f, from the given data points is to use additional significant interpolation conditions, i.e. to add an extra set of *off-surface points*. Once we define the augmented data set, we can then compute a three dimensional interpolant $\mathcal{I}$, via the PU method, to the total set of points [32, 75].

Thus, the reconstruction of the surface leads to a method consisting of three steps:

i. generate the extra off-surface points,

ii. find the interpolant of the augmented data set,

iii. render the iso-surface of the fit.

Let us suppose that, for each point $\boldsymbol{x}_i$, the oriented normal $\boldsymbol{n}_i \in \mathbb{R}^3$ is given. We construct the extra off-surface points by taking a small step away along the surface normals, i.e. we obtain for each data point $\boldsymbol{x}_i$ two additional off-surface points. One point lies *outside* the manifold $\mathcal{M}$ and is given by

$$\boldsymbol{x}_{N+i} = \boldsymbol{x}_i + \Delta \boldsymbol{n}_i,$$

whereas the other point lies *inside* $\mathcal{M}$ and is given by

$$\boldsymbol{x}_{2N+i} = \boldsymbol{x}_i - \Delta \boldsymbol{n}_i,$$

Δ being the stepsize. The union of the sets $\mathcal{X}_\Delta^+ = \{\boldsymbol{x}_{N+1}, \ldots, \boldsymbol{x}_{2N}\}$, $\mathcal{X}_\Delta^- = \{\boldsymbol{x}_{2N+1}, \ldots, \boldsymbol{x}_{3N}\}$ and $\mathcal{X}$, namely $\mathcal{X}_{3N}$, gives the overall set of points on which the interpolation conditions are assigned. Note that if we have zero normals in the given normal data set, we must exclude such points. Finally, we construct the augmented set of function values $\mathcal{F}_{3N}$. It is defined as the union of the following sets:

$$\begin{aligned}
\mathcal{F}_n &= \{f_i \ : \ f(\boldsymbol{x}_i) = a,\ i = 1, \ldots, n\}, \\
\mathcal{F}_\Delta^+ &= \{f_i \ : \ f(\boldsymbol{x}_i) = b,\ i = n+1, \ldots, 2n\}, \\
\mathcal{F}_\Delta^- &= \{f_i \ : \ f(\boldsymbol{x}_i) = c,\ i = 2n+1, \ldots, 3n\}.
\end{aligned}$$

Now, after creating the data set, we compute the interpolant $\mathcal{I}$ whose zero contour (iso-surface $\mathcal{I} = 0$) interpolates the given point cloud data.

The values $+1$ or -1 are arbitrary. Their precise value is not as critical as the choice of Δ. In fact the stepsize can be rather critical for a good surface fit [32]. Finally, we just render the resulting approximating surface $\mathcal{M}^*$ as the zero contour of the 3D interpolant [32]. If the normals are not explicitly given, we now illustrate some techniques to estimate them.

5.2.2 Normals estimation

To implement the implicit PU method, for each point $\boldsymbol{x}_i$, we need to find the oriented normal $\boldsymbol{n}_i$. To this aim, we follow the technique presented in [43, 44]. Of course, we have to assume that the surface is indeed orientable.

Given data of the form $\mathcal{X} = \{\boldsymbol{x}_i \in \mathbb{R}^3,\ i = 1, \ldots, N\}$, we fix a number $K < N$, and we find, for every point $\boldsymbol{x}_i$, the K nearest neighbors. The set of the neighbors of $\boldsymbol{x}_i$ is denoted by $\mathcal{K}(\boldsymbol{x}_i)$. The first step is to compute an oriented tangent plane for each data point [44]. The elements that describe the tangent plane $\mathcal{T}_p(\boldsymbol{x}_i)$ are a point $\boldsymbol{o}_i$, called the centre, and a unit normal vector $\boldsymbol{n}_i$. The latter is computed so that the plane is the least squares best fitting plane to $\mathcal{K}(\boldsymbol{x}_i)$. So, the centre $\boldsymbol{o}_i$ is taken to be the centroid of $\mathcal{K}(\boldsymbol{x}_i)$ and the normal $\boldsymbol{n}_i$ is determined using Principal Component Analysis (PCA), see e.g. [4, 45, 47].

More precisely, we compute the centre of gravity of $\{\boldsymbol{x}_k, k \in \mathcal{K}(\boldsymbol{x}_i)\}$, i.e.

$$\boldsymbol{o}_i = \frac{1}{K} \sum_{k \in \mathcal{K}(\boldsymbol{x}_i)} \boldsymbol{x}_k,$$

and the associated covariance matrix

$$\mathrm{Cov}(\boldsymbol{x}_i) = \sum_{k \in \mathcal{K}(\boldsymbol{x}_i)} (\boldsymbol{x}_k - \boldsymbol{o}_i)(\boldsymbol{x}_k - \boldsymbol{o}_i)^T,$$

which is a symmetric 3×3 positive semi-definite matrix. The eigenvalues $\lambda_{i1} \geq \lambda_{i2} \geq \lambda_{i3}$ and the corresponding unit eigenvectors $\boldsymbol{v}_{i1}, \boldsymbol{v}_{i2}, \boldsymbol{v}_{i3}$ of this positive semi-definite matrix represent the plane and the normal to this plane. Specifically, let us suppose that two eigenvalues λ_{i1} and λ_{i2} are close together and the third one is significantly smaller, so the eigenvectors for the first two eigenvalues $\boldsymbol{v}_{i1}$ and $\boldsymbol{v}_{i2}$ determine the plane, while the eigenvector $\boldsymbol{v}_{i3}$ determines the normal to this plane.

The second step is to orient the normal consistently, in fact $\boldsymbol{n}_i$ is chosen to be either $\boldsymbol{v}_{i3}$ or $-\boldsymbol{v}_{i3}$. Note that if two data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$ are close, their associated normals $\boldsymbol{n}_i$ and $\boldsymbol{n}_k$ are nearly parallel, i.e. $\boldsymbol{n}_i \boldsymbol{n}_k^T \approx \pm 1$. Consequently, if $\boldsymbol{n}_i \boldsymbol{n}_k^T \approx -1$ either $\boldsymbol{n}_i$ or $\boldsymbol{n}_k$ should be flipped. The difficulty in finding a consistent global orientation is that this condition should hold between all pairs of *sufficiently close* data points.

A common practice is to model this problem as graph optimization [44]. At first, we build the *Riemann graph* $G = \{\mathcal{V}, \mathcal{E}\}$, with each node in $\mathcal{V}$ corresponding to one of the 3D data points. We remark that the Riemann graph is defined as the undirect graph among which there exists an edge e_{ik} in $\mathcal{E}$ if v_k is one of the K nearest neighbors of v_i and vice versa. In our case, the graph G has a vertex for every normal $\boldsymbol{n}_i$ and an edge e_{ik} between the vertices of $\boldsymbol{n}_i$ and $\boldsymbol{n}_k$ if and only if $i \in \mathcal{K}(\boldsymbol{x}_k)$ or $k \in \mathcal{K}(\boldsymbol{x}_i)$. For example, to build a weighted graph, we could choose the weights $w(e_{ik}) = \boldsymbol{n}_i \boldsymbol{n}_k^T$. So the cost of the edge connecting the vertices $\boldsymbol{n}_i$ and $\boldsymbol{n}_k$ represents the deviation of the normals. Hence, the normals are consistently oriented if we find directions $b_i = \{-1, 1\}$, so that $\sum_{e_{ik}} b_i b_k w(e_{ik})$ is maximized. Unfortunately, this problem is NP-hard, i.e. no method can guarantee of finding its exact solution in a reasonable time, as shown in [43].

We propose the approximate solution described in [43]. The idea is simply to start with an arbitrary normal orientation and then to propagate it to neighboring normals. Intuitively, we would like to choose an order of propagation that favors propagation from $\mathcal{T}_p(\boldsymbol{x}_i)$ to $\mathcal{T}_p(\boldsymbol{x}_k)$ if the unoriented planes are nearly parallel. To assign orientation to an initial plane, the unit normal of the tangent plane whose centre has the third largest coordinate is made to point toward the positive x_3-axis. We assign to each edge e_{ik} the cost $w(e_{ik}) = 1 - |\boldsymbol{n}_i \boldsymbol{n}_k^T|$, as suggested by [43]. Note that $w(e_{ik})$ is small if the unoriented tangent planes are nearly parallel. A favourable propagation order can therefore be achieved by traversing the *minimal spanning tree* of the Riemann graph. The advantage of this order consists in propagating the orientation along directions of low curvature in the data.

To such scope, we need some preliminary definitions (see e.g. [3]) for further details.

Definition 17. *In any connected graph G, a spanning tree is a subgraph of G having the following two properties:*

i. the subgraph is a tree,

ii. the subgraph contains every vertex of G.

Definition 18. *The weight of a tree is defined to be the sum of the weights of all edges in the tree.*

Definition 19. *Given a connected weighted graph G the minimal spanning tree is the spanning tree having minimum weight among all spanning trees in the graph.*

We now want to determine how to construct a minimum weight spanning tree. To this aim, we use the Kruskal's algorithm (e.g. refer to [37] for further details). Precisely, we begin by choosing an edge of minimum weight in the graph and then we continue by selecting from the remaining edges an edge of minimum weight until a spanning tree is formed.

We now give an illustration of the implicit PU technique in a 2D setting [32].

Example 9. *Let us consider the following data set*

$$\boldsymbol{x}_i = \left([2 + \sin(t_i)] \cos(t_i), [2 + \cos(t_i)] \sin(t_i)\right), \quad i = 1, \dots, N,$$

where t_i is a Halton sequence.

Even if the normals can be analytically computed, in what follows we suppose that they are unknown.

Let us fix $N = 75$, see Figure 5.3.

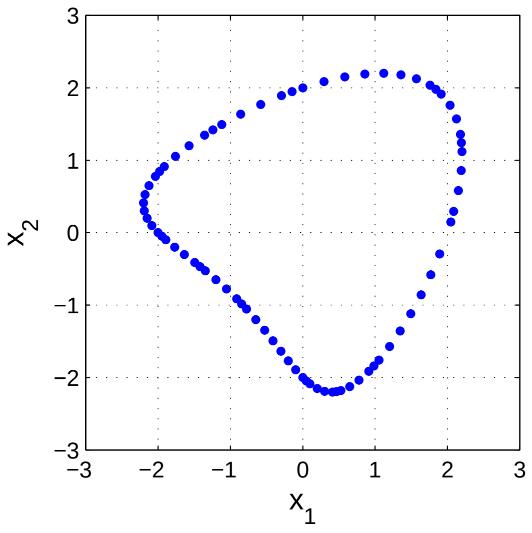


Figure 5.3: The point cloud data set.

To approximate the normals we use $K = 6$ nearest neighbors and then we propagate the orientation by traversing the minimal spanning tree of the Riemann graph, as shown in Figure 5.4 (left) and (right), respectively.

Next, to obtain the set of off-surface points, we add the function values. Specifically, we assign the value 0 to each original data point and the value 1 or -1 to outside or inside points (obtained by marching a small distance Δ along the normals), see Figure 5.5 (left). The step size is taken to be 1% of the size of the data set.

Now the problem is turned into a full 2D interpolation problem. Thus, in order to reconstruct the surface interpolating the augmented data set, we use the PU method with the Wendland's C^2 function $\varphi_{3,1}$. Moreover, the same function is used for the PU weights. We point out that in this dissertation the Wendland's C^2 function will be always used for the computation of the PU weights. The result, choosing the shape parameter ε equal to 0.6, is shown in Figure 5.5 (right).

The interpolant curve, shown in Figure 5.6, is the zero contour of the interpolant surface.

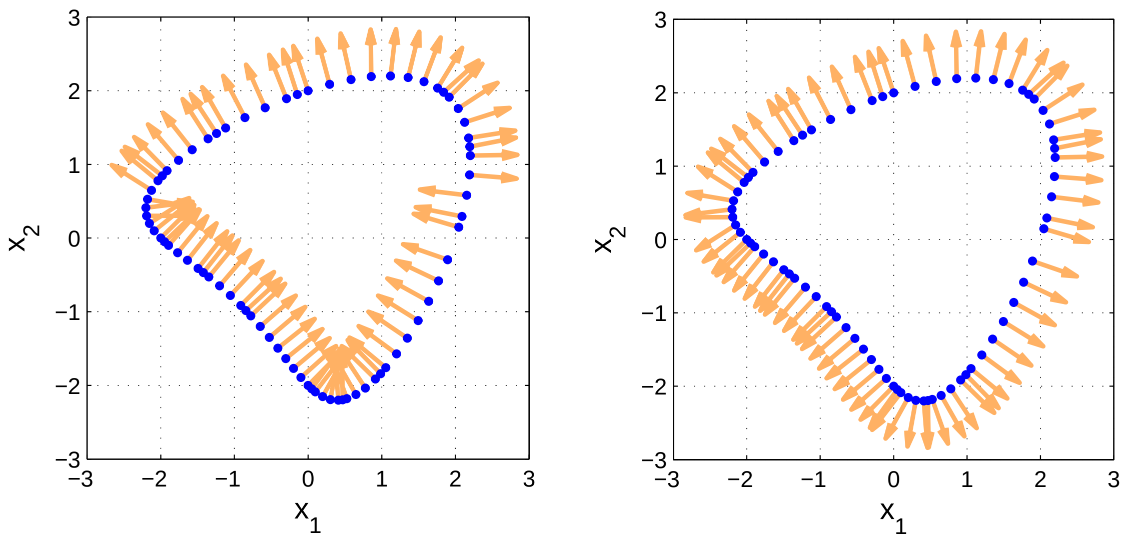


Figure 5.4: Inconsistent normals estimation (left) and consistent set of normals (right).

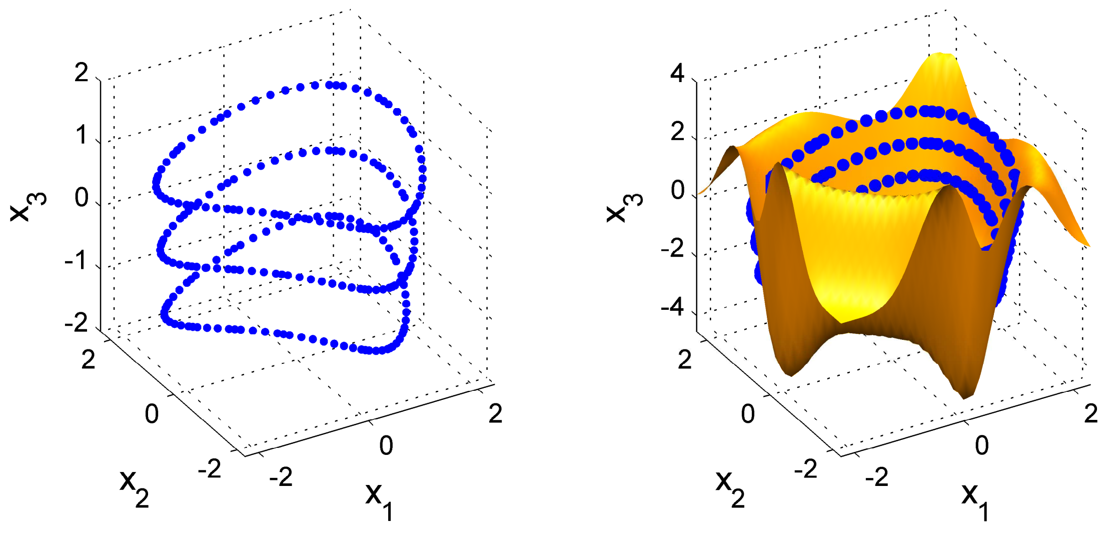


Figure 5.5: The augmented data set (left) and the surface interpolating the augmented data set (right).

5.3 Exercises

i. Prove that

$$|f(\boldsymbol{x}) - \mathcal{I}(\boldsymbol{x})| \leq \max_{i=1,\dots,M} ||f|_{\Omega_j} - P_{f_j}||_{L_\infty(\Omega_j)}.$$

ii. Construct a suitable multidimensional PU setting made of balls, i.e. choose suitable

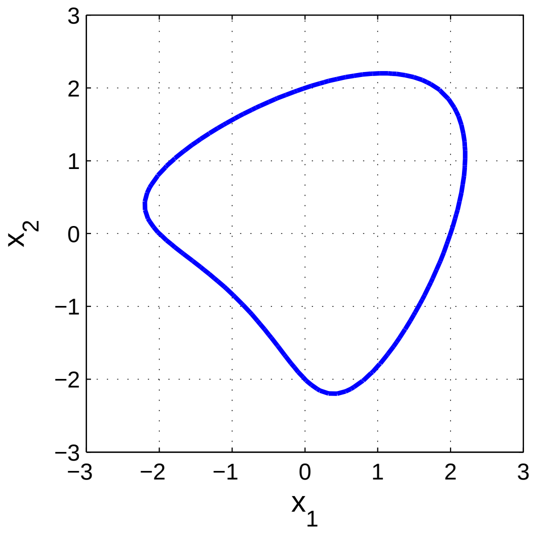


Figure 5.6: The interpolant curve, i.e. the zero contour of the interpolant surface.

PU centres and locations.

iii. If we use squares or rectangles instead of balls, what are the consequences on the PU weights? Hint: we need to modify the supports of the CSRBFs.

iv. Following [74], prove the theorem on the PU convergence.

Lecture 6

Collocation methods via RBFs

Collocation methods via RBFs for PDEs take advantage of being meshfree and easy to compute. In what follows, we consider symmetric collocation, Kansa's approach, local schemes and the method of lines.

6.1 RBF-collocation methods for PDEs

Given a linear elliptic differential operator $\mathcal{L}$, the goal consists in finding an approximate solution of the BVP problem (Dirichlet boundary conditions)

$$\begin{aligned} \mathcal{L}f(\boldsymbol{x}) &= g_1(\boldsymbol{x}), \qquad \text{for } \boldsymbol{x} \in \Omega, \\ f(\boldsymbol{x}) &= g_2(\boldsymbol{x}), \qquad \text{for } \boldsymbol{x} \in \partial\Omega. \end{aligned} \tag{6.1}$$

6.1.1 Kansa's Approach

We present here a method for computing the solution of elliptic BVPs via Kansa's collocation method which was introduced by E. Kansa in [48]. Originally it consisted of an unsymmetric scheme, based on multiquadrics, whose convergence properties were studied only later by R. Schaback (see e.g. [65]).

The problem (6.1) is discretized on a global set of collocation points $\mathcal{X} = \mathcal{X}_{N_c} \cup \mathcal{X}_{N_b} = \{\boldsymbol{x}_i, \ i = 1, \ldots, N\}$, where N_b and N_c are the number of nodes on $\partial\Omega$ and $\Omega \backslash \partial\Omega$, respectively.

For Kansa's collocation method we then choose to represent the approximate solution P_f by a radial basis function expansion analogous to that used for scattered data interpolation, i.e.,

$$P_f(\boldsymbol{x}) = \sum_{i=1}^{N} c_i \phi(||\boldsymbol{x} - \boldsymbol{\xi}_i||_2),$$

where we now formally distinguish in our notation between centers $\Xi_N = \{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_N\}$ and collocation points $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$. While formally different, these points will often physically coincide. For the following discussion we assume the simplest possible setting, i.e., no polynomial terms are added to the RBF expansion. The collocation matrix that arises will be of the form

$$L = \begin{pmatrix} A_L \\ A \end{pmatrix}$$

where

$$(A)_{ik} = \phi(||\boldsymbol{x}_i - \boldsymbol{\xi}_k||_2), \quad \boldsymbol{x}_i \in \partial\Omega, \quad \boldsymbol{\xi}_k \in \partial\Omega,$$

and

$$(A_L)_{ik} = \mathcal{L}\phi(||\boldsymbol{x} - \boldsymbol{\xi}_k||_2)|_{\boldsymbol{x}=\boldsymbol{x}_i}, \quad \boldsymbol{x}_i \in \Omega, \quad \boldsymbol{\xi}_k \in \Omega.$$

Finally, we need to solve the system

$$L\boldsymbol{c} = \boldsymbol{f},$$

where $\boldsymbol{c} = (c_1, \ldots, c_N)^T$ and $\boldsymbol{f} = (\boldsymbol{g}_1, \boldsymbol{g}_2)^T$.

Moreover, Kansa suggests the use of varying shape parameters. While the theoretical analysis of the resulting method is near intractable, Kansa shows that this technique improves the accuracy and stability of the method when compared to using only one constant value. A problem with Kansa's method is that the matrix L may be singular for certain configurations of the centers [42]. This fact is not really surprising since the matrix for the collocation problem is composed of rows that are built from different functions, which might not even be radial. Since the numerical experiments of Hon and Schaback show that Kansa's method cannot be well-posed for arbitrary center locations, it is now an open question to find sufficient conditions on the center locations that guarantee invertibility of the Kansa matrix. One possible approach is to adaptively select *good* centers from a large set of possible candidates. Following this strategy it is possible to ensure invertibility of the collocation matrix throughout the iterative algorithm. This approach is described in [49].

6.1.2 Hermite-based Approach

To introduce the symmetric approach, we need to recall the basic theory of Hermite-based interpolation. To this aim, let us take $\{\boldsymbol{x}_i, \lambda_i f\}$, $i = 1, \ldots, N$, where $\Lambda = \{\lambda_1, \ldots, \lambda_N\}$ is a linearly independent set of continuous linear functionals. For instance, λ_i could denote point evaluation at the points and thus we recover the interpolation conditions, or it could denote evaluation of some derivative at the points. In this setting, we also allow more general functionals, such as integrals.

In our context, we assume the generalized Hermite interpolant to be of the form

$$P_f(\boldsymbol{x}) = \sum_{i=1}^{N} c_i \lambda_i^{\boldsymbol{\xi}} \phi(||\boldsymbol{x} - \boldsymbol{\xi}||_2),$$

where the notation $\lambda_i^{\boldsymbol{\xi}}$ indicates that the functional acts on ϕ viewed as a function of its second argument. Furthermore, we do not add any superscript if λ acts on the kernel as a function of its first variable. Of course, similarly as we do for interpolation conditions, we require that

$$\lambda_i P_f = \lambda_i f, \quad i = 1, \ldots, N.$$

Therefore, after imposing such conditions, we reduce to solving:

$$A\boldsymbol{c} = \boldsymbol{f}_\lambda,$$

where $\boldsymbol{f}_\lambda = (\lambda_1 \boldsymbol{f}, \ldots, \lambda_N \boldsymbol{f})^T$ and the entries of the matrix are given by

$$(A)_{ik} = \lambda_i \lambda_k^{\boldsymbol{\xi}} \phi, \quad i, k = 1, \ldots, N.$$

It can be shown that A is non-singular for the same classes of ϕ that were admissible for the standard scattered data interpolation.

For the collocation setting, we propose the following expansion for the unknown solution of the PDE:

$$P_f(\boldsymbol{x}) = \sum_{k=1}^{N_c} c_k \mathcal{L}^{\boldsymbol{\xi}} \phi(||\boldsymbol{x} - \boldsymbol{\xi}||_2)_{\boldsymbol{\xi}=\boldsymbol{\xi}_k} + \sum_{k=N_c+1}^{N} c_k \phi(||\boldsymbol{x} - \boldsymbol{\xi}_k||_2).$$

After imposing the collocation conditions, we obtain:

$$L = \begin{pmatrix} A_{\mathcal{L}\mathcal{L}^{\boldsymbol{\xi}}} & A_{\mathcal{L}} \\ A_{\mathcal{L}^{\boldsymbol{\xi}}} & A \end{pmatrix}$$

where

$$(A_{\mathcal{L}\mathcal{L}^{\boldsymbol{\xi}}})_{ik} = \mathcal{L}\mathcal{L}^{\boldsymbol{\xi}} \phi(||\boldsymbol{x} - \boldsymbol{\xi}||_2)|_{\boldsymbol{x}=\boldsymbol{x}_i, \boldsymbol{\xi}=\boldsymbol{\xi}_k}, \quad \boldsymbol{x}_i, \boldsymbol{\xi}_k \in \Omega,$$
$$(A_{\mathcal{L}})_{ik} = \mathcal{L}\phi(||\boldsymbol{x} - \boldsymbol{\xi}_k||_2)|_{\boldsymbol{x}=\boldsymbol{x}_i}, \quad \boldsymbol{x}_i \in \Omega, \quad \boldsymbol{\xi}_k \in \partial\Omega$$
$$(A_{\mathcal{L}^{\boldsymbol{\xi}}})_{ik} = \mathcal{L}^{\boldsymbol{\xi}} \phi(||\boldsymbol{x}_i - \boldsymbol{\xi}||_2)|_{\boldsymbol{\xi}=\boldsymbol{\xi}_k}, \quad \boldsymbol{x}_i \in \partial\Omega, \quad \boldsymbol{\xi}_k \in \Omega$$
$$(A)_{ik} = \phi(||\boldsymbol{x}_i - \boldsymbol{\xi}_k||_2), \quad \boldsymbol{x}_i, \boldsymbol{\xi}_k \in \partial\Omega.$$

Thus, if the RBF centers coincide with collocation points we certainly have a symmetric matrix L and we have the following convergence result [74].

Theorem 18. *Let $\Omega \in \mathbb{R}^d$ be a polygonal and open region. Let $\mathcal{L} \neq 0$ be a second-order linear elliptic differential operator with coefficients in $C^{2(k-2)}(\bar{\Omega})$ that either vanish on $\bar{\Omega}$*

or have no zero there. Suppose that $\Phi \in C^{2k}(\mathbb{R}^d)$ is a strictly positive definite function. Suppose further that the boundary value problem

$$\begin{aligned} \mathcal{L}f(\boldsymbol{x}) &= g_1(\boldsymbol{x}), \qquad \textit{for } \boldsymbol{x} \in \Omega, \\ f(\boldsymbol{x}) &= g_2(\boldsymbol{x}), \qquad \textit{for } \boldsymbol{x} \in \partial\Omega, \end{aligned}$$

has a unique solution $f \in \mathcal{N}_\Phi(\Omega)$ for given $g_1 \in C(\Omega)$ and $g_2 \in C(\partial\Omega)$. Let P_f the approximate solution constructed via Hermite collocation, then

$$||f - P_f||_{L_\infty(\Omega)} \leq Ch^{k-2}||f||_{\mathcal{N}_\Phi(\Omega)},$$

for all sufficiently small h, where h is the larger of the fill distances in the interior and on the boundary of Ω, respectively.

6.1.3 Method of lines

Many PDEs of interest are time-dependent. Thus, in what follows we review a method for incorporating both spatial and time derivatives in the boundary value problems. To solve these time-dependent problems, we choose to use the method of lines (refer to [33]) because of its simplicity in dealing with the time component. For other time-dependent methods such as Laplace transforms see e.g. [70]. Let us consider

$$\frac{\partial}{\partial t} f(\boldsymbol{x}, t) = \mathcal{L}f(\boldsymbol{x}, t), \tag{6.2}$$

where at any fixed time t^*, the function

$$f^*(\boldsymbol{x}) = f(\boldsymbol{x}, t^*),$$

has only spatial arguments. In other words, the standard kernel-based methods can be used to approximate it. Let us introduce the following notation:

$$\boldsymbol{k}(\boldsymbol{x}) = (\phi(||\boldsymbol{x} - \boldsymbol{x}_1||_2), \dots, \phi(||\boldsymbol{x} - \boldsymbol{x}_N||_2)).$$

Then, we have that

$$P_f^*(\boldsymbol{x}) = \sum_{i=1}^{N} c_i^* \phi(||\boldsymbol{x} - \boldsymbol{x}_i||_2) = \boldsymbol{k}(\boldsymbol{x})^T \boldsymbol{c}^*.$$

where $\boldsymbol{c}^*$ must be solved in the interpolation context. Let us write the interpolation problem at any time t^* as

$$\begin{pmatrix} \mathcal{L}\boldsymbol{k}(\boldsymbol{x}_1)^T \\ \vdots \\ \mathcal{L}\boldsymbol{k}(\boldsymbol{x}_N)^T \end{pmatrix} \boldsymbol{c}^* = \begin{pmatrix} f^*(\boldsymbol{x}_1) \\ \vdots \\ f^*(\boldsymbol{x}_N) \end{pmatrix}.$$

Notice that if the kernel centres are fixed with respect to time, only $\boldsymbol{c}^*$ varies with the choice of time. This is the key factor which will allow us to separate the time and spatial components. Using the independence of the kernel centres with respect to time, we can write

$$P_f^*(\boldsymbol{x},t) = \boldsymbol{k}(\boldsymbol{x})^T \boldsymbol{c}(t).$$

Plugging this into the original PDE (6.2) gives

$$\boldsymbol{k}(\boldsymbol{x})^T \frac{\partial}{\partial t}\boldsymbol{c}(t) = \mathcal{L}\boldsymbol{k}(\boldsymbol{x})^T \boldsymbol{c}(t),$$

By collocating we obtain

$$\begin{pmatrix} \boldsymbol{k}(\boldsymbol{x}_1)^T \\ \vdots \\ \boldsymbol{k}(\boldsymbol{x}_N)^T \end{pmatrix} \frac{\partial}{\partial t}\boldsymbol{c}(t) = \begin{pmatrix} \mathcal{L}\boldsymbol{k}(\boldsymbol{x}_1)^T \\ \vdots \\ \mathcal{L}\boldsymbol{k}(\boldsymbol{x}_N)^T \end{pmatrix} \boldsymbol{c}(t).$$

In this way, we reduce to a system of N ODEs.

However, we need to incorporate the boundary conditions into the method of lines. Suppose that we consider

$$\begin{array}{rcccl} \frac{\partial}{\partial t} f(\boldsymbol{x},t) &=& \mathcal{L}f(\boldsymbol{x},t) + g_1(\boldsymbol{x}), & \boldsymbol{x} \in \Omega & t > 0, \\ 0 &=& f(\boldsymbol{x},t) + g_2(\boldsymbol{x}), & \boldsymbol{x} \in \partial\Omega & t \geq 0, \\ f(\boldsymbol{x},t) &=& f_0(\boldsymbol{x}), & \boldsymbol{x} \in \partial\Omega & t = 0, \end{array} \tag{6.3}$$

At first, we compute a kernel-based interpolant of the initial condition, which requires to solving

$$\begin{pmatrix} \boldsymbol{k}(\boldsymbol{x}_1)^T \\ \vdots \\ \boldsymbol{k}(\boldsymbol{x}_{N_c})^T \\ \boldsymbol{k}(\boldsymbol{x}_{N_c+1})^T \\ \vdots \\ \boldsymbol{k}(\boldsymbol{x}_{N_c+N_b})^T \end{pmatrix} \boldsymbol{c}(0) = \begin{pmatrix} f_0(\boldsymbol{x}_1) \\ \vdots \\ f_0(\boldsymbol{x}_{N_c}) \\ f_0(\boldsymbol{x}_{N_c+1}) \\ \vdots \\ f_0(\boldsymbol{x}_{N_c+N_b}) \end{pmatrix}.$$

Then, concerning the boundary we have

$$\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{k}(\boldsymbol{x}_{N_c+1})^T \\ \vdots \\ \boldsymbol{k}(\boldsymbol{x}_{N_c+N_b})^T \end{pmatrix} \boldsymbol{c}(t) + \begin{pmatrix} g_2(\boldsymbol{x}_{N_c+1}) \\ \vdots \\ g_2(\boldsymbol{x}_{N_c+N_b}) \end{pmatrix}.$$

For interior points we reduce to

$$\begin{pmatrix} \boldsymbol{k}(\boldsymbol{x}_1)^T \\ \vdots \\ \boldsymbol{k}(\boldsymbol{x}_{N_c})^T \end{pmatrix} \frac{\partial}{\partial t}\boldsymbol{c}(t) = \begin{pmatrix} \mathcal{L}\boldsymbol{k}(\boldsymbol{x}_1)^T \\ \vdots \\ \mathcal{L}\boldsymbol{k}(\boldsymbol{x}_{N_c})^T \end{pmatrix} \boldsymbol{c}(t) + \begin{pmatrix} g_1(\boldsymbol{x}_1) \\ \vdots \\ g_1(\boldsymbol{x}_{N_c}) \end{pmatrix}.$$

Thus, the full system of ordinary differential equation is given by

$$\begin{pmatrix} \boldsymbol{k}(\boldsymbol{x}_1)^T \\ \vdots \\ \boldsymbol{k}(\boldsymbol{x}_{N_c})^T \\ 0 \\ \vdots \\ 0 \end{pmatrix} \frac{\partial}{\partial t}\boldsymbol{c}(t) = \begin{pmatrix} \mathcal{L}\boldsymbol{k}(\boldsymbol{x}_1)^T \\ \vdots \\ \mathcal{L}\boldsymbol{k}(\boldsymbol{x}_{N_c})^T \\ \boldsymbol{k}(\boldsymbol{x}_{N_c+1})^T \\ \vdots \\ \boldsymbol{k}(\boldsymbol{x}_{N_c+N_b})^T \end{pmatrix} \boldsymbol{c}(t) + \begin{pmatrix} g_1(\boldsymbol{x}_1) \\ \vdots \\ g_1(\boldsymbol{x}_{N_c}) \\ g_2(\boldsymbol{x}_{N_c+1}) \\ \vdots \\ g_2(\boldsymbol{x}_{N_c+N_b}) \end{pmatrix},$$

and in this way we fully define the solution by means of the method of lines.

All the methods presented in above provide suitable solutions provided that we consider few nodes. Otherwise, it might be advantageous to work with the PU method.

6.1.4 Collocation via PU method

We present here a method for computing the solution of elliptic BVPs the partition of unity method.

Once we assume that (6.1) admits a solution of the form (5.1) then (see e.g. [64, 68]),

$$\begin{array}{rclcl} \mathcal{L}\mathcal{I}(\boldsymbol{x}_i) & = & \sum_{j=1}^{M} \mathcal{L}\left(W_j(\boldsymbol{x}_i) P_{f_j}(\boldsymbol{x}_i)\right) & = & g_1(\boldsymbol{x}_i), \quad \boldsymbol{x}_i \in \Omega \backslash \partial\Omega, \\ \mathcal{I}(\boldsymbol{x}_i) & = & \sum_{j=1}^{M} W_j(\boldsymbol{x}_i) P_{f_j}(\boldsymbol{x}_i) & = & g_2(\boldsymbol{x}_i), \quad \boldsymbol{x}_i \in \partial\Omega. \end{array} \tag{6.4}$$

Let $\boldsymbol{P}_{f_j} = (P_{f_j}(\boldsymbol{x}_1^j), \ldots, P_{f_j}(\boldsymbol{x}_{N_j}^j))^T$ be the vector of local nodal values. Since the local coefficients $\boldsymbol{c}_j = (c_1^j, \ldots, c_{N_j}^j)^T$ are so that $\boldsymbol{c}_j = A_j^{-1}\boldsymbol{P}_{f_j}$, we get

$$\mathcal{L}\boldsymbol{P}_{f_j} = A_j^{\mathcal{L}} A_j^{-1} \boldsymbol{P}_{f_j}, \tag{6.5}$$

where $A_j^{\mathcal{L}}$ is the matrix

$$(A_j^{\mathcal{L}})_{ik} := \mathcal{L}\phi(||\boldsymbol{x}_i^j - \boldsymbol{x}_k^j||_2), \quad i,k = 1,\ldots,N_j.$$

To obtain a discrete local operator L_j, we have to differentiate (6.4) by applying a product derivative rule and then use the relation (6.5).

To fix things, consider the Poisson problem, i.e. $\mathcal{L} = -\Delta$. The elliptic operator $\mathcal{L}$ can be expanded to get [41]

$$\begin{aligned} \mathcal{L}(W_j(\boldsymbol{x}_i)P_{f_j}(\boldsymbol{x}_i)) = & -\Delta W_j(\boldsymbol{x}_i)P_{f_j}(\boldsymbol{x}_i) - 2\nabla W_j(\boldsymbol{x}_i) \cdot \nabla P_{f_j}(\boldsymbol{x}_i) \\ & - W_j(\boldsymbol{x}_i)\Delta P_{f_j}(\boldsymbol{x}_i), \quad \boldsymbol{x}_i \in \Omega \backslash \partial\Omega, \end{aligned}$$

where the scalar product should be applied to the components of the gradients. Let A_j^{Δ} and A_j^{∇} be the matrices with entries

$$(A_j^{\Delta})_{ik} = \Delta\phi(||\boldsymbol{x}_i^j - \boldsymbol{x}_k^j||_2), \quad i,k = 1,\ldots,N_j,$$

and

$$(A_j^{\nabla})_{ik} = \nabla \phi(||\boldsymbol{x}_i^j - \boldsymbol{x}_k^j||_2), \quad i,k = 1,\ldots,N_j,$$

we have

$$\Delta \boldsymbol{P}_{f_j} = A_j^{\Delta} \boldsymbol{c}_j = A_j^{\Delta} A_j^{-1} \boldsymbol{P}_{f_j}.$$

Furthermore we consider the matrix

$$\bar{W}_j^{\Delta} = \text{diag}\left(\Delta W_j(\boldsymbol{x}_1^j), \ldots, \Delta W_j(\boldsymbol{x}_{N_j}^j)\right),$$

and similarly we define $\bar{W}_j^{\nabla}$ and $\bar{W}_j$. Finally, by including the boundary conditions, we can express the discrete local Laplacian as

$$(L_j)_{ik} = \begin{cases} (\bar{L}_j)_{ik}, & \text{for } \boldsymbol{x}_i^j \in \Omega \backslash \partial\Omega, \\ \delta_{ik}, & \text{for } \boldsymbol{x}_i^j \in \partial\Omega, \end{cases}$$

where δ_{ik} denotes the Kronecker delta and

$$\bar{L}_j = \left(\bar{W}_j^{\Delta} A_j + 2\bar{W}_j^{\nabla} \cdot A_j^{\nabla} + \bar{W}_j A_j^{\Delta}\right) A_j^{-1}. \tag{6.6}$$

In what follows we will refer to the collocation method described in this section as the RBF Standard approach (RBF-S), meaning that the standard basis is used to approximate the solution.

Note that, since we use the Laplacian, we require that both the kernel and the weight functions are at least twice differentiable. Let $\boldsymbol{x}_{\zeta_{kj}} \in \mathcal{X}_{N_j}$ be the node corresponding to $\boldsymbol{x}_k \in \mathcal{X}$. In order to obtain the global discrete PDE operator, we need to assemble the local ones into a global the matrix L as follows

$$(L)_{ik} = \sum_{j=1}^{M} (L_j)_{\zeta_{ij}.\zeta_{kj}}, \quad i,k = 1,\ldots,N.$$

Then, we simply have to solve the (possibly ill-conditioned) system

$$L\boldsymbol{z} = \boldsymbol{f}, \tag{6.7}$$

where $\boldsymbol{z} = (\mathcal{I}(\boldsymbol{x}_1), \ldots, \mathcal{I}(\boldsymbol{x}_N))^T$ and $\boldsymbol{f} = (f_1, \ldots, f_N)^T$, with

$$f_i = \begin{cases} g_1(\boldsymbol{x}_i), & \text{for } \boldsymbol{x}_i \in \Omega \backslash \partial\Omega, \\ g_2(\boldsymbol{x}_i), & \text{for } \boldsymbol{x}_i \in \partial\Omega, \end{cases} \quad i = 1,\ldots,N.$$

Remark. The main advantage of PU collocation is the computational efficiency in constructing the collocation matrix. However, we have to discuss several drawbacks concerning its well-posedness. In general, among meshfree global collocation methods, the symmetric ones should be preferred because they guarantee the existence and uniqueness of the solution. For Kansa's collocation approaches instead, the system might be singular [42] and its uniqueness can be ensured only under several restrictions which in particular lead to distinguish between collocation points and RBF centres [49]. The non-symmetry of the matrix L suggests that its non-singularity could be ensured only with a similar distinction between RBF centres and collocation data. This needs further investigations. Nevertheless, as for Kansa's collocation, the solution of PDEs via the PU method as presented here has already been shown to be an effective numerical tool.

Example 10. *In order to test the method, we consider an elliptic problem on $\Omega = [0,1]^2$ with a manufactured solution f from which we can compute g_1 and g_2. In particular [41]*

$$f(x_1,x_2) = \sin(x_1 + 2x_2^2) - \sin(2x_1^2 + (x_2 - 0.5)^2).$$

The graphical result is plotted in Figure 6.1.

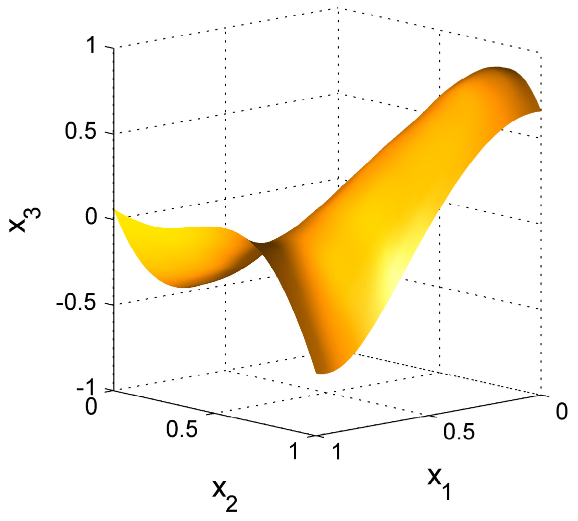


Figure 6.1: The approximate solution computed on $N = 289$ Halton data with the Gaussian C^∞ kernel as local approximant.

6.2 Exercises

i. Extend Kansa's collocation to conditionally positive definite functions.

ii. Explicitly write the collocation matrices for the Hermite-based approach when the Gaussian and Inverse MultiQuadric are used.

iii. Extend Kansa's collocation to elliptic equations with variable coefficients and homogeneous Dirichlet boundary conditions

$$\frac{\partial}{\partial x_1}\left(a(x_1,x_2)\frac{\partial}{\partial x_1}f(x_1,x_2)\right) + \frac{\partial}{\partial x_2}\left(b(x_1,x_2)\frac{\partial}{\partial x_2}f(x_1,x_2)\right) = f(x_1,x_2),$$

for $(x_1,x_2) \in \Omega = [0,1]^2$ and

$$f(x_1,x_2) = 0, \quad (x_1,x_2) \in \partial\Omega.$$

Lecture 7

Financial applications

In this chapter we present and discuss an RBF method for pricing financial contracts by solving the well-known *Black-Scholes* equation (in the sequel we simply refer to it as BS equation).

7.1 Introduction and notation

Financial markets have become more and more complex, with the trading of many different finacial products. Among them *derivatives* are very important and worth to be studied. Henceforth it is of fundamental interest to dispose of method that simulates and updates the financial products very fast and as accurately as possible (cf. [26, 63]).

Definition 20. *A derivative is a contract that derives its value from the performance of an underlying entity (for example: asset, index, interest rate), simply called* `underlying`

Here we introduce a small vocabulary (ordered alphabetically) of the terms that we will use.

Asset is an economical resourse convertible in cash, like stocks and obligations.

Basket options are options on a basket of assets. The assets can be stocks, commodities, indices and other financial securities. A commonly traded basket option is a *vanilla call/put option* which is a linear combination of assets.

Derivative is every contract whose value depend on underlyings (stocks, indexes, currencies, interest rates etc.). Common derivates are *forward, features, options, swaps* and their variation.

Payoff literarly means the final outcome or a result of an activity. In finance is given by difference between exercise price and spot price of contract.

Risk-free interest rate is the rate of return of a hypothetical investment with no risk of financial loss, over a given period of time.

Spot price is the price of goods traded on a spot market and available for almost immediate delivery.

Strike price is the price (or *exercise price*) at which a specific derivative contract can be exercised. The term is mostly used to describe stock and index options in which strike prices are fixed in the contract. For *call options*, the strike price is where the security can be bought (up to the expiration date); for *put options*, the strike price is the price at which shares can be sold.

Underlying is any entity (not necessarily financial) from which the derivative depends on.

Volatility (indicated often with σ) is the degree of variation of a trading price series over time as measured by the standard deviation of logarithmic returns.

The first problem we afford is pricing, by a model, a financial contract based on several underlyings (assets). The contract chosen is the European Basket Option. The model used in the well-known *Blank-Scholes-Merton* linear parabolic partial differential equation, in which the spatial dimension is determined by the number of assets.

For completeness, there exist other models for pricing high-dimensional contracts like Monte-Carlo, sparse grids or finite differences.

7.2 The multi-dimensional Black-Scholes model

The BS equation (cf. e.g. [62]), is a linear parabolic PDE that models the price evolution of an *European call option* (or an European put option). We start our discussion with the one-dimensional case.

Let $F(s,t)$ be the price of the option at the time t while s indicates the space variable which represent the *stock price*. The BS equation determines the arbitrage free price of a contigent claim and is given as

$$\begin{cases} \dfrac{\partial F}{\partial t} + r\, s\, \dfrac{\partial F}{\partial s} + \dfrac{1}{2}\sigma^2 s^2 \dfrac{\partial^2 F}{\partial s^2} - rF = 0 \\ \\ F(s,T) = \Psi(s) \end{cases} \tag{7.1}$$

where σ is the *volatity* and r the *risk-free interest rate*. The equation domain is the stripe $s > 0, t \in [0,T)$. By solving this equation backwards in time, we obtain the arbitrage free price of the contingent claim for all s we are interested in.

Following [62], the multi-dimensional version of equation (7.1), is

$$\begin{cases} \dfrac{\partial F}{\partial t} + r \displaystyle\sum_{i=1}^{d} s_i \frac{\partial F}{\partial s_i} + \frac{1}{2} \sum_{i,j=1}^{d} [\sigma \sigma^*]_{i,j}\, s_i s_j \frac{\partial^2 F}{\partial s_i \partial s_j} - rF = 0 \\ \\ F(\mathbf{s}, T) = \Psi(\mathbf{s}) \end{cases} \tag{7.2}$$

Here d is the number of the underlyings, $\mathbf{s} = (s_1, \ldots, s_d)^T$ and σ is now a matrix of volatities ($[\sigma\sigma^*]$ is symmetric and positive definite). We can transform (7.2) in a non-dimensional initial value problem by using the transformations (scaling)

$$\begin{array}{l} \mathbf{s} = K\mathbf{x}, \;\; \bar{r} = r/\hat{\sigma}^2, \;\; \hat{t} = \frac{1}{2}\hat{\sigma}^2(T-t)\,, \\ P(\mathbf{x}, \hat{t}) = F(\mathbf{s}, t)/K, \;\; \bar{\sigma} = \sigma/\hat{\sigma}, \;\; K\Phi(\mathbf{x}) = \Psi(\mathbf{s}) \end{array} \tag{7.3}$$

with the constant $\hat{\sigma} = \max_{ij}\{\sigma_{ij}\}$, we get the dimensionless equation

$$\begin{cases} \dfrac{\partial P}{\partial \hat{t}} = 2\bar{r} \displaystyle\sum_{i=1}^{d} x_i \frac{\partial P}{\partial x_i} + \sum_{i,j=1}^{d} [\bar{\sigma}\bar{\sigma}^*]_{i,j}\, x_i x_j \frac{\partial^2 P}{\partial x_i \partial x_j} - 2\bar{r}P \\ \\ P(\mathbf{x}, 0) = \Phi(\mathbf{x}) \end{cases} \tag{7.4}$$

Letting

$$\mathcal{L} := 2\bar{r} \sum_{i=1}^{d} x_i \frac{\partial}{\partial x_i} + \sum_{i,j=1}^{d} [\bar{\sigma}\bar{\sigma}^*]_{i,j}\, x_i x_j \frac{\partial^2}{\partial x_i \partial x_j} - 2\bar{r}$$

the elliptic operator, then the initial value problem (7.4) can be written

$$\begin{cases} \dfrac{\partial P}{\partial \hat{t}}(\mathbf{x}, \hat{t}) = \mathcal{L}P(\mathbf{x}, \hat{t}), \quad \hat{t} \in \mathbb{R}_+, \;\; \mathbf{x} \in \mathbb{R}^d_+ \\ P(\mathbf{x}, 0) = \Phi(\mathbf{x}) \end{cases} \tag{7.5}$$

In the case of the European call option the payoff function is

$$\Phi(\mathbf{x}) = \left(\frac{1}{d} \sum_{i=1}^{d} x_i - \tilde{K} \right)_+ = \max\left(\frac{1}{d} \sum_{i=1}^{d} x_i - \tilde{K}, 0 \right). \tag{7.6}$$

and due to the scalings (7.3), $\tilde{K} = 1$.

7.2.1 Which reference solution and in which domain?

As reference solution we can consider the one give by an *adaptive finite difference* method, as the one derived in [62]. For a finite difference discretization of BS, we need boundary conditions all around all the boundary. The implementation in [62] (and used also in [72]) assume that on all the boudary

$$\frac{\partial^2 P(\mathbf{x},\hat{t})}{\partial x_i^2} = 0 \tag{7.7}$$

which implies that the option price is nearly linear w.r.t. the spot price at the boundary. Notice that the previous conditions are equivalent to that of the derivatives along the normals, because of the shape of the domain (a stripe).

Inside the domain instead, we consider a second order finite difference approximation on a structured but not equispaced grid (i.e. adaptive). In formulas this can be written as follows

$$\frac{dP_h}{d\hat{t}} = A_h P_h$$

where P_h is the vector of ordered unknowns and A_h is the second order FD-discretizazion of the elleptic operator $\mathcal{L}$.

Unfortunately, the condtions (7.7) do not work well with RBF, especially with infinitely smooth kernels. In any case the problem is well-posed if the growth of the solution to infinity is limited.

For this reason we propose to use only of **near** and **far** field boundary conditions. This means that on the boundaries of type

$$\Gamma_i = \{\mathbf{x} \in \mathbb{R}^d_+ : \ \mathbf{x} \neq \mathbf{0}, \ x_i = 0\}, \quad i = 1, \ldots, d$$

- **Near field** is the condition at $\mathbf{x} = \mathbf{0}$: $P(\mathbf{0}, \hat{t}) = 0$;
- **Far field** we assume the asymptotic behavior of the solution

$$P(\mathbf{x},\hat{t}) \approx \frac{1}{d}\sum_{i=1}^{d} x_i - \bar{K}\mathsf{e}^{-2\bar{r}\hat{t}}, \quad \|\mathbf{x}\| \to \infty\,.$$

Obviously other choices can be made as suggested for instance in [31, 26].

Which computational domain?

- The hypercube

$$D = \bigotimes_{i=1}^{d} [0, a_i]$$

  for constructing the structured grid for the FD approximation.
- RBF approach is meshfree. This suggests to take *artificial* far-field boundary conditions at our choice.

Concerning the RBF method, considering the payoff function (7.6), it makes sense to use the **boundary surface**

$$\sum_{i=1}^{d} x_i = C$$

where the constant C will be chosen far enough from the origin so that the far-field solution, introduced above, will be a good approximation of the our solution.

7.2.2 Computing the error

As for every differential equation depending on the time, solving the BS equation we want to know the solution at the time $\hat{t} = T$.

1. Practically, this means to know the price of an option today with excercise time T years from today. The error at a generic point $\mathbf{x}$ will then be
$$e(\mathbf{x}) = P(\mathbf{x}, T) - u(\mathbf{x}, T)\,. \tag{7.8}$$
2. The second observation to make is that the most interesting region (called *Region of Interest (RoI)*) where to consider the error is when the *mean value of the stock prices is in a neighbor of the strike price.* This means that
$$\frac{1}{d}\sum_{i=1}^{d} x_i \in U_{\bar{K}} \tag{7.9}$$
with $U_{\bar{K}}$ an interval around $\bar{K}$ that will be given by the observation of the referred stock market. For instance, in [63], observing the Stockholm stock exhange market they suggested (see also Fig. 7.1) $U_{\bar{K}} = \left[\frac{\bar{K}}{3}, \frac{5\bar{K}}{3}\right]$

On the basis of these two obesrvations we can consider as *financial error*

$$E_f = \max_{\mathbf{x}\in U_{\bar{K}}} |e(\mathbf{x})| \tag{7.10}$$

Another financial norm that can be used is the following *weighted intergral norm* (cf. [62])

$$E_w = \int_{\Omega} |e(\mathbf{x})| w(\mathbf{x}) d\mathbf{x} \tag{7.11}$$

where Ω is the whole domain and the weight function is chosen as a product of d Gaussians centered in the RoI and such that $\int_\Omega w(\mathbf{x})d\mathbf{x} = 1$. For example in 1-d, $w_1(x) \approx \exp(-5(x - \bar{K})^2)$ amd in 2-d $w_2(x_1, x_2) \approx \exp(-4(x_1 + x_2 - 2\bar{K})^2)\exp(-(x_1 - x_2)^2)$. In the case of the normalized value $\bar{K} = 1$ we get $\displaystyle\int_0^2 w_1(x)dx \approx 0.79$ and $\displaystyle\int_{[0,2]^2} w_2(\mathbf{x})d\mathbf{x} \approx 0.77$, see Fig. 7.2 for the plot of these two functions. These functions are indeed good approximation of the finantial error in the RoI.

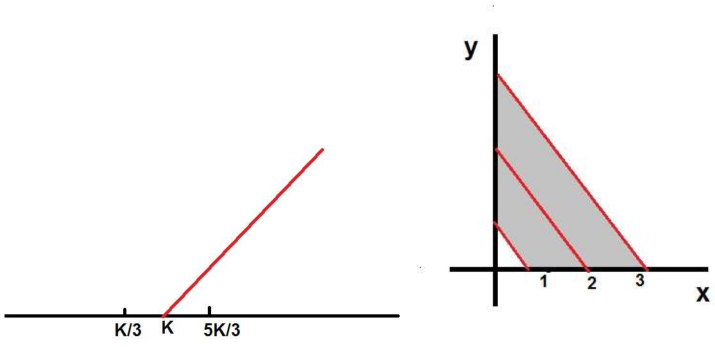


Figure 7.1: Region of Interest 1d and 2d

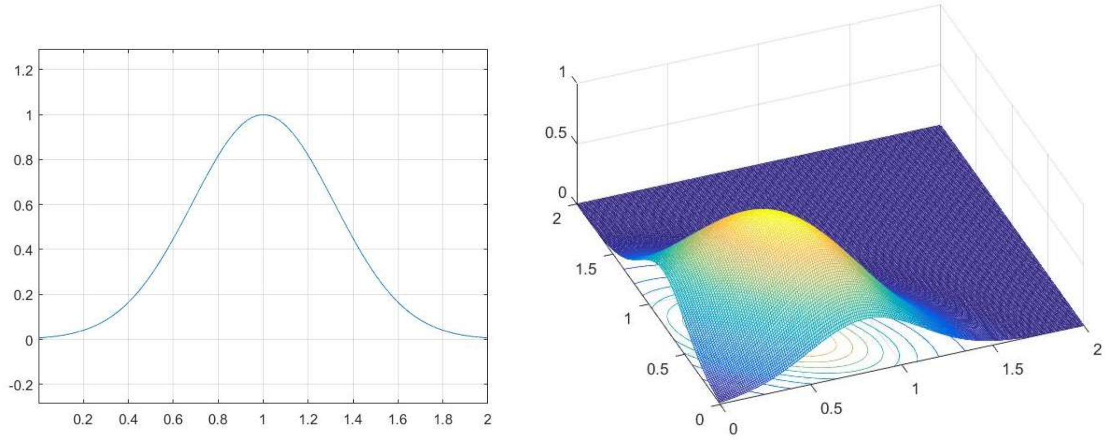


Figure 7.2: Plot of the weight functions w_1 and w_2

7.2.3 RBF approximation and time-stepping

Our approximation in time is a linear combination of radial basis functions. Notice that in the sequel for the sake of simplicity we use t intending $\hat{t}$.

$$u(\mathbf{x},t) = \sum_{i=1}^{N} \lambda_i(t)\phi(\epsilon(\|\mathbf{x}-\mathbf{x}_i\|) = \sum_{i=1}^{N} \lambda_i(t)\phi_i(\mathbf{x})\,. \tag{7.12}$$

At the time t the coefficients λ_i are determined by collocation, as follows.

- **Interior nodes** $x_i,\ i = 1,\dots,N_i$ we use the PDE
$$\frac{\partial u(\mathbf{x},t)}{t} = \mathcal{L}u(\mathbf{x},t)\,.$$
- **Near and far-field nodes** $x_i,\ i = N_i+1,\dots,N$ we impose $u(\mathbf{0},t) = 0$ and $u(\mathbf{x},t) \approx \frac{1}{d}\sum_{j=1}^{d} x_j - \bar{K}e^{-2\bar{r}t}$ as $\|x\| \to \infty$.

Now defining the vectors $\mathbf{u}_i(t) = (u(\mathbf{x}_1, t), \ldots, u(\mathbf{x}_{N_i}, t))^T$ and $\mathbf{u}_b(t) = (u(\mathbf{x}_{N_i+1}, t), \ldots, u(\mathbf{x}_N, t))^T$ we can write the linear system

$$\begin{pmatrix} \mathbf{u}_i(t) \\ \mathbf{u}_b(t) \end{pmatrix} = \underbrace{\begin{pmatrix} A_{ii} & A_{ib} \\ A_{bi} & A_{bb} \end{pmatrix}}_{A} \begin{pmatrix} \boldsymbol{\lambda}_i(t) \\ \boldsymbol{\lambda}_b(t) \end{pmatrix} \tag{7.13}$$

where the whole matrix A has entries $a_{ij} = \phi(\epsilon\|\mathbf{x}_i - \mathbf{x}_j\|)$. If the radial function $\phi$is positive definite, we know that the matrix is invertible. Moreover

$$\mathcal{L}\mathbf{u}_i(t) = \underbrace{(B_{ii}\ \ B_{ib})}_{B} \begin{pmatrix} \boldsymbol{\lambda}_i(t) \\ \boldsymbol{\lambda}_b(t) \end{pmatrix} = (B_{ii}\ \ B_{ib})\, A^{-1} \begin{pmatrix} \mathbf{u}_i(t) \\ \mathbf{u}_b(t) \end{pmatrix} = \underbrace{(C_{ii}\ \ C_{ib})}_{C} \begin{pmatrix} \mathbf{u}_i(t) \\ \mathbf{u}_b(t) \end{pmatrix} \tag{7.14}$$

with B the rectangular matrix with entries $b_{jk} = \mathcal{L}\phi(\epsilon\|\mathbf{x}_j - \mathbf{x}_k\|), \quad j = 1, \ldots, N_i, \ k = 1, \ldots, N$.

As well known, the stability of the time-steps is determined by the eigenvalues of the block C_{ii} which *all have no positive real part.* This allows to use, for the advancing time step an *unconditionally stable* method like BDF2.

Let $t_n = c\,n$ be our constant time step and $\mathbf{u}_i^n \approx \mathbf{u}_i(t_n)$. Then the BDF2 applied to our problem is

$$\mathbf{u}_i^n + b_1\mathbf{u}_i^{n-1} + b_2\mathbf{u}_i^{n-2} = c\, b_0\mathcal{L}\mathbf{u}_i^n \tag{7.15}$$

where the coefficients b_s are $b_0 = 1, b_1 = -1, b_2 = 0$ for the initial step and $b_0 = 2/3, b_1 = -4/3, b_2 = 1/3$ for the successive steps.

At each time step the boundary conditions are $\mathbf{u}_b^n = \boldsymbol{\gamma}_b^n$ with $\boldsymbol{\gamma}_b^n = (\gamma(\mathbf{x}_{N_i+1}, t^n), \ldots, \gamma(\mathbf{x}_N, t^n))^T$ and the function γ is such that

$$\gamma(\mathbf{x}, t) = \begin{cases} 0 & \mathbf{x} = \mathbf{0} \\ \frac{1}{d}\sum_{k=1}^{d} x_k - \bar{K}e^{-2\bar{r}t} & \|x\|_1 = C \end{cases}$$

Now, combining (7.14), (7.15) and the boundary conditions $\mathbf{u}_b^n$ we get the advancing time step scheme

$$\begin{pmatrix} I - c\, b_0\, C_{ii} & -c\, b_0\, C_{i,b} \\ 0 & I \end{pmatrix} \begin{pmatrix} \mathbf{u}_i^n \\ \mathbf{u}_b^n \end{pmatrix} = \begin{pmatrix} -b_1\mathbf{u}_i^{n-1} - b_2\mathbf{u}_i^{n-2} \\ \boldsymbol{\gamma}_b^n \end{pmatrix} \tag{7.16}$$

and the initial conditions vector $\mathbf{u}_i^0 = (\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_{N_i}))^T$.

7.2.4 Heurist error analysis: node distribution and shape parameter

The aim is to reduce the error in the RoI. Therefore, chosen the model parameters (that is ϕ, $\bar{r}$, $\bar{\sigma}$, the final time step M so that we get the exercise time T) and the far-field surface C, then the accuracy of the method would depend on *node distribution* and the *shape parameter*.

In the paper [63] the considered for the examples that we are going to show the following data

- $\phi(r) = \sqrt{1+r^2}$, multiquadrics
- far-field boundary surface $\dfrac{1}{d}\sum_{i=1}^{d} x_i = 4\bar{K}$
- $\bar{r} = 5/9$ (corresponding to $r = 0.05$)
- $\bar{\sigma} = 1$ (corresponding to $\sigma = 0.3$) while in the 2d case $\bar{\sigma} = \begin{pmatrix} 1 & 1/6 \\ 1/6 & 1 \end{pmatrix}$ corresponding to $\sigma = \begin{pmatrix} 0.3 & 0.05 \\ 0.05 & 0.3 \end{pmatrix}$
- the maximum M so that $M+1$ does not improve the accuracy.
- exercise time $T = 0.045$ which corresponds to 1 year.

About the node distribution, we can use an adaptive deterministic strategy based on the fact that we want to reduce the finantial error in the RoI. The recipe is simple. Let us consider the 1d case first. For some $p \in \mathbb{N}$, we can distribute for example $N = 3p+2$ points as follows: $p+1$ equally spaced points in the intervals $[0, \bar{K}-\delta]$ and $[\bar{K}+\delta, 2*\bar{K}]$, while the remaining p, also equally spaced, in $(\bar{K}-\delta, \bar{K}+\delta)$, with $\delta = 1/(N-1)$ (the spacing). Similarly for the 2d case. An example for $p = 5$ is displayed in Fig. 7.3. As shown in [63]

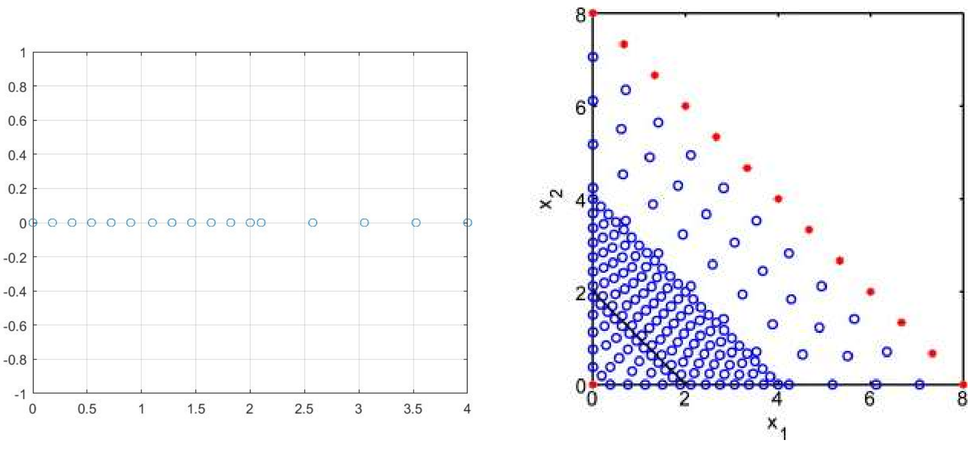


Figure 7.3: node distribution in 1d and 2d for $p = 5$

this strategy reduces significantly the finantial error E_f. The price to pay is of a bigger condition number of A, but we know that this is the a consequence of the trade-off principle.

Another way to reduce the error is by chosing the *optimal* shape parameter, say ϵ^*, by one of the strategies discussed in Section 4.6.

7.2.5 Accuracy in space and time

- RBF provide spectral accuracy when the ϕ has infinite smoothness. In order to see this accuracy in space one has to fix ϵ and varying $N = 20...200$ compute the error (this is the non-stationary approach). One can observe that indeed one get

$$E_f = O(e^{-\alpha N}) \ \texttt{with} \ \alpha = O(10^{-2})$$

- For accuracy in time, we fix N and ϵ, but we vary the time-steps, say from $M = 2, \ldots, 10^3$. Since we have used a BDF2 discretization method, we can expect to have and order 2 of convergence as required. We can also observe that using the weighted norm it is possible to reduce further the error.

This method, proposed in [63], compared with the adaptive FD, has shown to be 20 uo to 40 times faster in the low to intermediate accuracy range. The memory requirements still the same. The only limitation was that it was applied on maximum dimension 2.

7.3 A recent approach

A new **RBF-PUM** for pricing 1d and 2d *vanilla basket options* has been proposed in [68].

We recall that a vanilla call/put option is a basket option made of a linear combination of assets. Precisely, let $S_i(t),\ i = 1, ..., N$ the prices of N stocks at the time t. Consider

$$U(t) = \sum_{i=1}^{N} a_i S_i(t), \ \ i = 1, ..., N \ .$$

where the a_i are constants. A vanilla is an option on $U(t)$. In more detail, on an exercise date t, the payoff of the option is

$$\Phi(t) = (\alpha(U(t) - K)_+$$

with K the exercise price and $\alpha = 1$ for a call option or $\alpha = -1$ in the case of put.

In the case $N = 2, a_1 = 1, a_2 = -1$ the basket option is a *spread option.*

The method proposed is a *penalty* method for American basket call options. American call options differ from European call ones because they consider the dividend. In the case of put options the dividend may be zero.

The BS considered has a *penalty term*

$$P(V) = \frac{e\left(rK - \sum_{i=1}^{N} \alpha_i D_i x_i\right)}{V + e - q} \tag{7.17}$$

where e is the penalty parameter (chosen quite small), D_i is the dividend of the i-th assets, r the risk-free interest rate and

$$q(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i x_i - K$$

is the *barrier function* corresponding to the nonzero part of the payoff function Φ. The penalized BS equation, with initial and boundaty conditions is then

$$\begin{aligned}
\frac{\partial V}{\partial t} &= \mathcal{L}V - P(V), \;\; \mathbf{x} \in \Omega_E = \mathbb{R}^d_+, \;\; 0 < t \le T \\
V(\mathbf{x}, 0) &= \Phi(\mathbf{x}), \quad \mathbf{x} \in \Omega_E \\
V(\mathbf{0}, t) &= \mathbf{0}, \quad t \in (0, T] \\
V(\mathbf{x}, t) &= \Phi(\mathbf{x}), \quad \mathbf{x} \in \Gamma^E, \; t \in (0, T]\,.
\end{aligned}$$

This equation can be solved by RBF-PUM in space and BDF2 in time. Some attention should be done to the choice of the penalty parameter, the shape parameter and the number of partions. For details and numerical results see again [68].

7.4 Reference website

For interested people, at the link `http://www.it.uu.se/research/scientific_computing/project/compfin` one of the authors of the cited papers [63, 68], has made available more papers and software for computational finance application using RBF. In particular at the link `http://www.it.uu.se/research/scientific_computing/project/rbf/software/rbfpu_amop_penalty` is available the software for making experiments with the RBF-PUM for the approach described in the previous section.

Lecture 8

Software for PU

In what follows we list the codes for both PU interpolation and collocation.

We give a brief description of the implemented Matlab functions. For more details concerning input and output parameters and the usage of single functions, see the the comments within the Matlab routines.

8.1 Softwares for PU interpolation

```
1.  PU.m: computes the PU interpolant
2.  DistanceMatrix.m: computes the distance matrix of two
    sets of points
3.  MakeSdGrid.m: computes grid points
4.  IntegerBased_MD_ContainingQuery: used to organize
    centres of the subdomains
5.  IntegerBased_MD_Neighbourhood: finds the neighbourhood
6.  IntegerBased_MD_Structure: organizes points
7.  IntegerBased_MD_RangeSearch: finds the points
    lying on a subdomain
8.  LaplaceMain.m: computes the RBF-PU collocation
9.  W2weight.m: construct the PU weight functions
10. PU_weight.m: evaluates the PU weight functions
11. RBF_PUM_diffmat_gauss.m: it constructs
    differentiation matrices
12. gauss_PUM.m: it evaluates differentiation matrices


Remarks:
a. the function DistanceMatrix.m is from the book
   Meshfree Approximations Methods with Matlab,
   Fasshauer, World Scientific, 2007.
b. routines 4--7 are from
```

```
   http://hdl.handle.net/2318/158790
c. Routines 9--12 have been modified and are from
   http://www.it.uu.se/research/scientific_computing/
   project/rbf/software/rbfpum_convdiff

%-------------------------------------------------------------------%
%
% File: PU(M,dsites,neval,npu,rbf,wf,f,rhs,ep);
%
% Goal: script that performs partition of unity with variable
%       patches and shape parameters
%
% Inputs:  M:            space dimension
%          dsites:       NXM matrix representing a set of N data
%                        sites
%          neval:        evaluation points in one direction
%          npu:          PU subdomains in one direction
%          rbf:          radial basis function
%          wf:           weight function
%          f:            the test function
%          rhs:          the function values
%          ep:           the shape parameter
%
% Outputs:  epoints: the evaluation points
%           Pf:      the interpolant at the evaluation points
%
% Calls on: IntegerBased_MD_Structure,
%           IntegerBased_MD_Neighbourhood
%           IntegerBased_MD_RangeSearch,
%           IntegerBased_MD_ContainingQuery,
%           DistanceMatrix, MakeSDGrid.
%
% Remark:   DistanceMatrix, MakeSDGrid come from the book:
%           [G. E. Fasshauer, Meshfree approximation methods with
%           Matlab, World Scientific, Singapore, 2007].
%
%-------------------------------------------------------------------%
function [epoints,Pf] = PU(M,dsites,neval,npu,rbf,wf,f,rhs,ep)
% Create npu^M equally spaced PU centres
puctrs = MakeSDGrid(M,npu);
% Create neval^M equally spaced evaluation points
epoints = MakeSDGrid(M,neval);
puradius = 1./npu;  % Define the initial PU radius
wep = 1./puradius;  % Parameter for weight function
npu_M = size(puctrs,1); neval_M = size(epoints,1); % Initialize;
rbfctrs = dsites; % Define the RBF centres
Pf = zeros(neval_M,1);  % Initialize
% Compute Shepard evaluation matrix
DM_eval = DistanceMatrix(epoints,puctrs);
SEM = wf(wep,DM_eval);
SEM = spdiags(1./(SEM*ones(npu_M,1)),0,neval_M,neval_M)*SEM;
```

```
% Parameter for integer-based partitioning structure
q = 1./puradius;
% Build the partitioning structure for data and evaluation points
idx_ds = IntegerBased_MD_Structure(dsites,q,puradius,M);
idx_ep = IntegerBased_MD_Structure(epoints,q,puradius,M);
for j = 1:npu_M
puradius = 1/npu;
index1 = IntegerBased_MD_ContainingQuery(puctrs(j,:),q,...
puradius,M);
% Find data sites located in the j-th subdomain
[dxx,dx] = IntegerBased_MD_Neighbourhood(dsites,idx_ds,...
index1,q,M,1);
idx = IntegerBased_MD_RangeSearch(puctrs(j,:),puradius,...
dxx,dx);
[edxx,edx] = IntegerBased_MD_Neighbourhood(epoints,...
idx_ep,index1,q,M,1);
% Compute the distance matrix
DM_data = DistanceMatrix(dsites(idx,:),rbfctrs(idx,:));
% Compute the interpolation matrix
IM = rbf(ep,DM_data);
% Find the evaluation points located in the j-th subdomain
eidx = IntegerBased_MD_RangeSearch(puctrs(j,:),puradius,...
edxx,edx);
if  (~isempty(eidx))
% Compute evaluation matrix and the RBF interpolant
DM_eval = DistanceMatrix(epoints(eidx,:),rbfctrs(idx,:));
EM = rbf(ep,DM_eval); localfit = EM * (IM\rhs(idx));
% Accumulate global fit
Pf(eidx) = Pf(eidx) + localfit.*SEM(eidx,j);
end
end
% Compute exact solution
exact = f(epoints);
% Compute errors on evaluation grid
maxerr = norm(Pf - exact,inf);
rms_err = norm(Pf - exact)/sqrt(neval_M);
fprintf('RMS␣error:␣␣␣␣␣␣␣%e\n', rms_err);
fprintf('Maximum␣error:␣␣␣%e\n', maxerr);

%-----------------------------------------------------------------%
%
% File: DistanceMatrix(dsites,ctrs)
%
% Goal: Forms the distance matrix of two sets of points in R^M,
%
% Inputs: dsites: NxM matrix representing a set of N data in R^M
%                 (i.e., each row contains a M-dimensional point)
%         ctrs:   sxM matrix representing a set of s centers in
%                 R^M (one center per row)
%
% Outputs:  DM: Mxs matrix whose i,j position contains the
%                distance between the i-th data  and j-th center
%
%-----------------------------------------------------------------%
function DM = DistanceMatrix(dsites,ctrs)
[M,~] = size(dsites); [N,s] = size(ctrs);
DM = zeros(M,N);
% Accumulate sum of squares of coordinate differences
% The ndgrid command produces two MxN matrices:
%   dr, consisting of N identical columns (each containing
%       the d-th coordinate of the M data sites)
%   cc, consisting of M identical rows (each containing
%       the d-th coordinate of the N centers)
for d=1:s
[dr,cc] = ndgrid(dsites(:,d),ctrs(:,d));
DM = DM + (dr-cc).^2;
end
DM = sqrt(DM);

%-----------------------------------------------------------------%
%
% File: MakeSDGrid(s,neval)
%
% Goal: Produces matrix of equally spaced points
%
% Inputs: s:     space dimension
%         neval: number of points in each coordinate direction
%
% Outputs: gridpoints: neval^s-by-s matrix
%
%-----------------------------------------------------------------%
function gridpoints = MakeSDGrid(s,neval)
if (s==1)
gridpoints = linspace(0,1,neval)';
return;
end
% Mimic this statement for general s:
% [x1, x2] = ndgrid(linspace(0,1,neval));
outputarg = 'x1';
for d = 2:s
outputarg = strcat(outputarg,',x',int2str(d));
end
makegrid = strcat('[',outputarg,']=ndgrid(linspace(0,1,neval));');
eval(makegrid);
% Mimic this statement for general s:
% gridpoints = [x1(:) x2(:)];
gridpoints = zeros(neval^s,s);
for d = 1:s
matrices = strcat('gridpoints(:,d)␣=␣x',int2str(d),'(:);');
eval(matrices);
end
```

```
%-------------------------------------------------------------------%
%
% File: IntegerBased_MD_ContainingQuery(puctr,q,puradius,M)
%
% Goal: script that given a subdomain centre returns the index of
%       the square block containing the subdomain centre
%
% Inputs: puctr:        subdomain centre
%         q:            number of blocks in one direction
%         puradius:     radius of the PU subdomains
%         M:            space dimension
%
% Outputs: index: the index of the block containing the subdomain
%                 centre
%
%-------------------------------------------------------------------%
function [index] = IntegerBased_MD_ContainingQuery(puctr,q,...
puradius,M)
idx = ceil(puctr./puradius); k = 1:M-1;
idx(idx == 0) = 1;
index = sum((idx(k)-1).*q.^(M-k)) + idx(end);

%-------------------------------------------------------------------%
%
% File: IntegerBased_MD_Neighbourhood(dsites,idx_ds,index1,q,M,t)
%
% Goal: script that founds the neighbouring blocks
%
% Inputs:  dsites:  NXM matrix representing a set of N data sites
%          idx_ds:  the integer-based data structure
%          index1:  indices of points lying in the k-th block
%          q:       number of blocks in one direction
%          M:       space dimension
%          t:       it is related to the number of neighbouring
%                   blocks, the procedure searches in the k-th block
%                   and in its (3 + 2^t)^M - 1 neighboring blocks
%
% Outputs:  dxx: points lying in the k-th neighbourhood
%           dx:  indices of points lying in the k-th neighbourhood
%
%-------------------------------------------------------------------%
function [dxx, dx] = IntegerBased_MD_Neighbourhood(dsites,...
idx_ds,index1,q,M,t)
neigh = []; k = M-1; index = index1; k1 = 1:t; % Initialize
% Find neighbouring blocks
while k  > 0
neigh = [index1+k1.*q.^k,index1-k1.*q.^k];
if k - 1 > 0
neigh = [neigh,neigh+q.^(k-1),neigh-q.^(k-1)];
end
k = k - 1;
```

```
end
k2 = 1:t; k3 = k2;
for i = 1:length(neigh)
neighplus(k2) = neigh(i) + k3;
neighminus(k2) = neigh(i) - k3;
k2 = k2 + t;
end
neigh = [neigh,index1+k1,index1-k1,neighplus,neighminus];
% Reduce the number of neighbouring blocks for border blocks
j = find(neigh > 0 & neigh <= q^M);
index = [index; neigh(j)'];
dxx = []; dx = [];
for p = 1:length(index)
dxx = [dxx;dsites(idx_ds{index(p)},:)];
dx = [dx;idx_ds{index(p)}];
end

%-----------------------------------------------------------------%
%
% File: IntegerBased_MD_Structure(dsites,q,puradius,M)
%
% Goal: find the data sites located in each of the q^M blocks
%
% Inputs: dsites:      NXM matrix representing a set of N data sites
%         q:           number of blocks in one direction
%         puradius:    radius of PU subdomains
%         M:           space dimension
%
% Outputs: idx_dsites_k: multiarray containing the indices of the
%                        points located in k-th block
%
%-----------------------------------------------------------------%
function [idx_dsites_k] = IntegerBased_MD_Structure(dsites,q,...
puradius,M)
N = size(dsites,1); idx_dsites_k = cell(q^M,1); k = 1:M-1;
for i = 1:N
idx = ceil(dsites(i,:)./puradius);
idx(idx == 0) = 1;
index = sum((idx(k)-1).*q.^(M-k)) + idx(end);
idx_dsites_k{index} = [idx_dsites_k{index}; i];
end

%-----------------------------------------------------------------%
%
% File: IntegerBased_MD_RangeSearch(puctr,puradius,dsites,index)
%
% Goal: find the data sites located in a given subdomain and the
%       distances between the subdomain centre and data sites
%
% Inputs: puctr:     subdomain centre
%         puradius:  radius of PU subdomains
%          dsites:    NXM matrix representing a set of N data sites
%          index:     vector of the indices of the data points
%                     located in the k-th block
%
% Outputs: idx:  vector containing the indices of the data points
%                located in a given PU subdomain
%           dist: vector containing the distances between the data
%                sites and the subdomain centre
%
%-------------------------------------------------------------------%
function [idx, dist] = IntegerBased_MD_RangeSearch(puctr,...
puradius,dsites,index)
N = size(dsites,1); dist = []; idx = []; % Initialize
% Compute distances between the data sites and the centre
for i = 1:N
dist1(i) = norm(puctr - dsites(i,:));
end
% Use a sort procedure to order distances
if N > 0
[sort_dist,IX] = sort(dist1);
N1 = size(sort_dist,2); j1 = 1; j2 = 1; %Initialize
% Find the data sites located in the given subdomain
if nargin == 3
idx = IX; dist = dist1;
else
while (j2 <= N1) && (sort_dist(j2) <= puradius)
idx(j1) = index(IX(j2)); dist(j1) = dist1(IX(j2));
j1 = j1 + 1; j2 = j2 + 1;
end
end
end
```

8.2 Softwares for PU collocation

```
%-------------------------------------------------------------------%
%
% File: LaplaceMain(N,x,neval,ep,fun,Lfun,phi)
%
% Goal: script that solves the Poisson equation by means of HVSKs
%
% Inputs: N:        number of central points
%         x:        the central points
%         neval:    number of evaluation points in one direction
%         ep :      shape parameter
%         fun:      the test function
%         Lfun:     Laplacian of the test function
%         phi:      the kernel function
%
%-------------------------------------------------------------------%
function [RMSE] = LaplaceMain(N,x,neval,ep,fun,Lfun,phi)
```

```
% Define the evauation points
grid = linspace(0,1,neval); [xee,yee] = meshgrid(grid);
epoints = [xee(:) yee(:)]; Ne = size(epoints,1);
% Define boundary nodes
sn = sqrt(N); bdylin = linspace(0,1,sn)';
bdy0 = zeros(sn-1,1); bdy1 = ones(sn-1,1);
xb = [bdylin(1:end-1) bdy0; bdy1 bdylin(1:end-1);...
flipud(bdylin(2:end)) bdy1; bdy0 flipud(bdylin(2:end))];
% Define the global set of nodes (central and boundary points)
dsites = [xb;x];
% The number of patches in one dimension
npu =  ceil(sqrt(N)/2);
% Define the PU centres
xp = linspace(0,1,npu); cellradius = sqrt(2)/npu;
[xp,yp] = meshgrid(xp);
puctrs = [xp(:) yp(:)];
npu = size(puctrs,1); Nb = size(xb,1); N = size(dsites,1);
% Parameter for integer-based partitioning structure
q = ceil(1./cellradius);
% Build the partitioning structure for data and evaluation points
idx_ds = IntegerBased_MD_Structure(dsites,q,cellradius,2);
idx_ep = IntegerBased_MD_Structure(epoints,q,cellradius,2);
for i=1:npu
% Find the block cointaining the j-th subdomain centre
index = IntegerBased_MD_ContainingQuery(puctrs(i,:),q,...
cellradius,2);
% Find data sites located in the j-th subdomain
[dxx,dx] = IntegerBased_MD_Neighbourhood(dsites,idx_ds,...
index,q,2,1);
idx = IntegerBased_MD_RangeSearch(puctrs(i,:),cellradius,...
dxx,dx);
locpts(i).ind = sort(idx);
[edxx,edx] = IntegerBased_MD_Neighbourhood(epoints,...
idx_ep,index,q,2,1);
eidx = IntegerBased_MD_RangeSearch(puctrs(i,:),cellradius,...
edxx,edx);
elocpts(i).ind = sort(eidx);
Ni(i) = length(locpts(i).ind); % Initialize
end
% Define the pu weight
pu = W2weight(dsites,locpts,puctrs,cellradius);
epu = W2weight(epoints,elocpts,puctrs,cellradius);
B = spalloc(N,N,sum(Ni.^2)); % Initialize
for i = 1:npu
idx = locpts(i).ind; % Compute points on the j-th subdomain
% Compute the interpolation matrix
[A,Ax,Axx,Ay,Ayy] = RBF_PUM_diffmat_gauss(dsites(idx,:),...
ep);
A_L = Axx+Ayy;
% Construct the local Laplacian
D_L = diag(pu(i).w)*(A_L) + ...
```

```
2*diag(pu(i).wx)*(Ax) + 2*diag(pu(i).wy)*(Ay) + ...
diag(pu(i).wxx + pu(i).wyy)*(A);
% Accumulate into the global collocation matrix
B(idx,idx) = B(idx,idx) + D_L/A;
% Compute the evaluation points on the j-th subdomain
eidx = elocpts(i).ind;
if (~isempty(eidx))
E(i).Ae = phi(ep,DistanceMatrix(epoints(eidx,:),...
dsites(idx,:)));
E(i).Ae = diag(epu(i).w)*E(i).Ae/A;
else
E(i).Ae = zeros(0,length(idx));
end
end
% Organize boundary conditions
B(1:Nb,1:N) = eye(Nb,N); f = fun(xb(:,1),xb(:,2));
g = Lfun(x(:,1),x(:,2)); rhs = [f;g];
% Solve the system
u = B\rhs;
% Compute exact solution and RMSE
ue  = zeros(Ne,1);
for i=1:npu
idx=locpts(i).ind;
eidx = elocpts(i).ind;
ue(eidx) = ue(eidx) + E(i).Ae*u(idx);
end
exact = fun(epoints(:,1),epoints(:,2));
RMSE = norm(exact-ue,2)/neval;

%-----------------------------------------------------------------%
%
% File: W2weight(xc,locpts,Cp,cellradius)
%
% Goal:  solves the Poisson equation by means of stable methods
%
% Inputs: xc:          the collocation points
%         locpts:      the collocation points on the subdomains
%         Cp:          centre of PU subdomains
%         cellradius: the radius of patches
%
% Outputs: pu: the PU weights
%
% Calls on: PU_weight
%
%-----------------------------------------------------------------%
function [pu] = W2weight(xc,locpts,Cp,cellradius)
Np = size(Cp,1); % Initialize
% Construct the weight functions
[phi,phix,phiy,phixx,phiyy] = PU_weight(xc,locpts,Cp,cellradius);
% Compute the sums of the generating functions and their derivatives
s = sum(phi,2); sx = sum(phix,2); sy = sum(phiy,2); ...
```

```
sxx = sum(phixx,2);
syy = sum(phiyy,2);
% Compute the weight for the PU collocation
for i = 1:Np
loc = locpts(i).ind;
if isempty(loc)
loc = find([0;0;0]); s(loc)=find([0;0;0]);
end
pu(i).w = phi(loc,i)./s(loc);
pu(i).wx = phix(loc,i)./s(loc) - ...
phi(loc,i).*sx(loc)./s(loc).^2;
pu(i).wy = phiy(loc,i)./s(loc) - ...
phi(loc,i).*sy(loc)./s(loc).^2;
pu(i).wxx = -2*phix(loc,i).*sx(loc)./s(loc).^2 + ...
phixx(loc,i)./s(loc) +...
phi(loc,i).*(2*sx(loc).^2./s(loc).^3 - ...
sxx(loc)./s(loc).^2);
pu(i).wyy = -2*phiy(loc,i).*sy(loc)./s(loc).^2 + ...
phiyy(loc,i)./s(loc) + ...
phi(loc,i).*(2*sy(loc).^2./s(loc).^3 - ...
syy(loc)./s(loc).^2);
end

%-------------------------------------------------------------------%
%
% File: PU_weight(xc,locpts,Cp,cellradius)
%
% Goal: evaluates Wendland's function and its derivatives
%
% Inputs: xc:          the collocation points
%         locpts:      the collocation points on the subdomains
%         Cp:          centre of PU subdomains
%         cellradius: the radius of patches
%
% Outputs: phi,phix,phiy,phixx,phiyy: evaluation of the PU weights
%                                      and the derivatives
%
%-------------------------------------------------------------------%
function [phi,phix,phiy,phixx,phiyy] = PU_weight(xc,locpts,Cp,...
cellradius)
Np = size(Cp,1);
for i = 1:length(Np)
N(i) = length(locpts(i).ind);
end
zs = spalloc(size(xc,1),Np,sum(N));
[phi,phix,phiy,phixx,phiyy] = deal(zs);
for i = 1:Np
rho = 1/cellradius;
dx = xc(locpts(i).ind,1)-Cp(i,1);
dy = xc(locpts(i).ind,2)-Cp(i,2);
r = sqrt(dx.^2 + dy.^2);
```

```
r1 = r.*rho;
phi(locpts(i).ind,i) = (4*rho*r+1).*((1-rho*r)).^4;
phix(locpts(i).ind,i) = -20*rho^2*dx.*((1-rho*r)).^3;
phiy(locpts(i).ind,i) = -20*rho^2*dy.*((1-rho*r)).^3;
if r == 0
phixx=20/cellradius.^2; phiyy=20/cellradius.^2;
else
phixx(locpts(i).ind,i)=20*((1-r1)).^2.*(-r1+4*...
(dx.^2./cellradius.^2)+(dy.^2./cellradius.^2))./...
cellradius.^2./r1;
phiyy(locpts(i).ind,i)=20*((1-r1)).^2.*(-r1+(dx.^2....
/cellradius.^2)+4*(dy.^2./cellradius.^2))./...
cellradius.^2./r1;
end
end

%-------------------------------------------------------------------%
%
% File: RBF_PUM_diffmat_gauss(data,ep)
%
% Goal: script that computes the differentiation matrices
%
% Inputs: data:       the collocation points
%         ep:         the shape parameter
%
% Calls on: gauss_PUM
%
% Outputs: A,Ax,Axx,Ay,Ayy: the differentian matrices
%
% Calls on: gauss_PUM
%
%-------------------------------------------------------------------%
function [A,Ax,Axx,Ay,Ayy] = RBF_PUM_diffmat_gauss(data,ep)
N = size(data,1); [A,Ax,Ay,Axx,Ayy] = deal(zeros(N,N));
% Generate differentiation matrices
for i = 1:N
[psi,psix,psiy,psixx,psiyy] = gauss_PUM(ep,data,data(i,:));
% Compute the differentiation matrices
A(i,:) = psi; Ax(i,:) = psix; Ay(i,:) = psiy; Axx(i,:) = psixx;
Ayy(i,:) = psiyy;
end

%-------------------------------------------------------------------%
%
% File: gauss_PUM(ep,xc,xe)
%
% Goal: script that evaluates the differentiation matrices
%
% Inputs: ep:         the shape parameter
%         xc:         the data sites
%         xe:         the evaluation points
```

```
%
% Outputs: w,wx,wy,wxx,wyy: evaluation of the gaussian function and
%                           its the derivatives
%
%------------------------------------------------------------------%
function [w,wx,wy,wxx,wyy] = gauss_PUM(ep,xc,xe)
dx = xe(1,1) - xc(:,1); dy = xe(1,2) - xc(:,2); % Initialize
r2 = dx.^2 + dy.^2; % Initialize
% Evaluate the RBF and its derivatives
we = exp(-ep^2*r2); w=we';
wex = (-2*ep^2*dx).*we; wx=wex';
wey = (-2*ep^2*dy).*we; wy=wey';
wexx =(-2*ep^2).*we+(4*dx.^2*ep^4).*we; wxx=wexx';
weyy =(-2*ep^2).*we+(4*dy.^2*ep^4).*we; wyy=weyy';
```

8.3 Exercises

i. Run the following code

```
%------------------------------------------------------------------%
close all
clear all
warning off
disp('-------------------------------------------------------')
M = 2;                                   % space dimension
N = 17.^M;                               % number of scattered data
dsites1 = haltonset(M);                  % the type of data
dsites = dsites1(1:N,:);
neval = 40;                              %  for evaluation points
npu = floor(((N)./(4))^(1/M));           % parameter for PU centres
rbf = @(ep, r) exp(-(ep*r).^2);          % local kernel type
wf = @(ep,r) (max(1-(ep*r),0).^4).* ...
(4*(ep*r)+1);                          % define weight function
f = @(x) (x(:,1)+x(:,2)-1).^9;           % the test function
rhs = f(dsites);                         % the function values
ep = 1;                                  % the shape parameter
PU(M,dsites,neval,npu,rbf,wf,f,rhs,ep);
```

ii. Run the following code

```
%------------------------------------------------------------------%
close all
clear all
warning off
disp('-------------------------------------------------------')
N = 81; % Number of central nodes
% Define the central nodes
load('data_demo');
```

```
neval = 40; % Number of evaluation points
ep = 1; % Define the shape parameters
% The test function and its laplacian
fun = @(x1,x2) sin(x1.^2+2*x2.^2)-sin(2*x1.^2+(x2-0.5).^2);
Lfun = @(x1,x2) 6*cos(x1.^2+2*x2.^2)-6*cos((x2-1/2).^2+2*x1.^2) ...
-4*x1.^2.*sin(x1.^2+2*x2.^2)-16*x2.^2.*sin(x1.^2+2*x2.^2) ...
+sin((x2-1/2).^2+2*x1.^2).*(2*x2-1).^2 ...
+ 16*x1.^2.*sin((x2-1/2).^2+2*x1.^2);
phi = @(ep,r) exp(-ep.^2.*r.^2); % The RBF
RMSE = LaplaceMain(N,x,neval,ep,fun,Lfun,phi);
fprintf('RMSE                        %e \n',RMSE);
```

iii. Repeat the exercises with the Inverse MultiQuadrics.

Bibliography

[1] G. Allasia, R. Besenghi, R. Cavoretto, A. De Rossi, *Scattered and track data interpolation using an efficient strip searching procedure*, Appl. Math. Comput. **217** (2011), pp. 5949–5966.

[2] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, A.Y. Wu, *An optimal algorithm for approximate nearest neighbor searching in fixed dimensions*, J. ACM **45** (1998), pp. 891–923.

[3] M. Behzad, G. Chartrand, L. Lesniak-Foster, *Graphs and Digraphs*, Prindle, Weber and Schmidt, Boston, 1979.

[4] D. Belton, *Improving and extending the information on principal component analysis for local neighborhoods in 3D point clouds*, in: C. Jun et al. (Eds.), The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-Part B5, vol. 37, 2008, pp. 477–484.

[5] S. Bochner, *Vorlesungen über Fouriersche Integrale*, Akademische Verlagsgesellschaft, Leipzig, 1932.

[6] L. Bos, M. Caliari, S. De Marchi, M. Vianello, Y. Xu, *Bivariate Lagrange interpolation at the Padua points: the generating curve approach*, J. Approx. Theory **143** (2006), pp. 15–25.

[7] I. Babuška, J.M. Melenk, *The partition of unity method*, Int. J. Numer. Meth. Eng. **40** (1997), pp. 727–758.

[8] M.D. Buhmann, *Radial Basis Functions: Theory and Implementation*, Cambridge Monogr. Appl. Comput. Math., vol. 12, Cambridge Univ. Press, Cambridge, 2003.

[9] J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. Mccallum, T.R. Evans, *Reconstruction and representation of 3D objects with radial basis functions*, in: Proceedings of the 28-th Annual Conference on Computer Graphics and Interactive Techniques, ACM press, New York, 2001, pp. 67–76.

[10] J.C. Carr, W.R. Fright, R.K. Beatson, *Surface interpolation with radial basis functions for medical imaging*, IEEE Trans. Med. Imaging **16** (1997), pp. 96–107.

[11] R. Cavoretto, *A numerical algorithm for multidimensional modeling of scattered data points*, Comput. Appl. Math. **34** (2015), pp. 65–80.

[12] R. Cavoretto, A. De Rossi, *Achieving accuracy and efficiency in spherical modelling of real data*, Math. Method. Appl. Sci. **37** (2014), pp. 1449–1459.

[13] R. Cavoretto, A. De Rossi, *Spherical interpolation using the partition of unity method: An efficient and flexible algorithm*, Appl. Math. Lett. **25** (2012), pp. 1251–1256.

[14] R. Cavoretto, A. De Rossi, *Fast and accurate interpolation of large scattered data sets on the sphere*, J. Comput. Appl. Math. **234** (2010), pp. 1505–1521.

[15] R. Cavoretto, A. De Rossi, E. Perracchione, *Efficient computation of partition of unity interpolants through a block-based searching technique*, Comput. Math. Appl. **71** (2016), pp. 2568–2584.

[16] Y.L. Chen, S.H. Lai, *A partition of unity based algorithm for implicit surface reconstruction using belief propagation*, in: IEEE International Conference on Shape Modeling and Applications 2007, IEEE Press, Los Alamitos, 2007, pp. 147–155.

[17] K.C. Chung, T.H. Yao, *On lattices adimmitting unique Lagrange interpolations*, SIAM J. Numer. Anal. **14** (1977), pp. 735–743.

[18] I.K. Crain, B.K. Bhattacharyya, *Treatment of non-equispaced two-dimensional data with a digital computer*, Geoexploration **5** (1967), pp. 173–194.

[19] S. Cuomo, A. Galletti, G. Giunta, A. Starace, *Surface reconstruction from scattered point via RBF interpolation on GPU*, in: M. Ganzha et al. (Eds.), 2013 Federated Conference on Computer Science and Information Systems, IEEE Press, Los Alamitos, 2013, pp. 433–440.

[20] M. De Berg, M. Van Kreveld, M. Overmars, O. Schwarzkopf, *Computational Geometry*, Springer, Berlin, 1997.

[21] E.L. De Forest, *Additions to a memoir on methods of interpolation applicable to the graduation of irregular series*, Annual Report of the Board of Regents of the Smithsonian Institution for 1873, pp. 319–353, 1874.

[22] E.L. De Forest, *On some methods of interpolation applicable to the graduation of irregular series*, Annual Report of the Board of Regents of the Smithsonian Institution for 1871, pp. 275–339, 1873.

[23] C. de Boor, *A Practical Guide to Splines*, revised edition, Springer, New York 2001.

[24] S. De Marchi, On Leja sequences: some results and applications, Appl. Math. Comput. **152** (2004), pp. 621–647.

[25] S. De Marchi, R. Schaback, H. Wendland, *Near-Optimal Data-independent Point Locations for Radial Basis Function Interpolation* , Adv. Comput. Math. **23** (2005), pp. 317–330

[26] Stefano De Marchi, Maddalena Mandarà and Anna Viero, *Meshfree approximation of multi-asset European and American option problems*, Aracne ed. Roma, pp. 76, ISBN: 9788854851511 (2012).

[27] J. Duchon, *Sur l'erreur d'interpolation des fonctions de plusieurs variables par les D^m-splines*, Rev. Francaise Automat. Informat. Rech. Opér. Anal. Numer. **12** (1978), pp. 325–334.

[28] J. Duchon, *Splines minimizing rotation-invariant semi-norms in Sobolev spaces*, in: W. Schempp at al. (Eds.), Constructive Theory of Functions of Several Variables, Springer Lecture Notes in Mathematics, Springer-Verlag, Berlin, vol. 571, 1977, pp. 85–100.

[29] J. Duchon, *Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces*, Rev. Francaise Automat. Informat. Rech. Opér. Anal. Numer. **10** (1976), pp. 5–12.

[30] N. Dyn, D. Levin, *Iterative solution of systems originating from integral equations and surface interpolation*, SIAM J. Numer. Anal. **2** (1981), pp. 377–390.

[31] G.E. Fasshauer, A.Q.M Khaliq and D.A. Voss, *Using Meshfree Approximations for multi-asset American Option Problems* , J. Chin. Inst. Eng. 27 (2004), pp. 563–571.

[32] G.E. Fasshauer, *Meshfree Approximations Methods with* Matlab, World Scientific, Singapore, 2007.

[33] G.E. Fasshauer, M.J. McCourt, *Kernel-based Approximation Methods Using* Matlab, World Scientific, Singapore, 2015.

[34] R. Franke, *Scattered data interpolation: Tests of some methods*, Math. Comp. **38** (1982), pp. 181–200.

[35] R. Franke, *A critical comparison of some methods for interpolation of scattered data*, Report No. NPS-53-79-003, Naval Postgraduate School, Monterey, 1982.

[36] W.R. Goodin, G.J. McRae, J.H. Seinfeld, *A comparison of interpolation methods for sparse data: Application to wind and concentration fields*, J. Appl. Meteor. **18** (1979), pp. 761–771.

[37] R.J. Gould, *Graph Theory*, Dover Publications, Mineola, 2012.

[38] J.P. Gram, *Über Entwicklung reeler Functionen in Reihen mittelst der Methode der kleinsten Quadrate*, J. Math. **94** (1883), pp. 41–73.

[39] R.L. Harder, R.N. Desmarais, *Interpolation using surface splines*, J. Aircraft **9** (1972), pp. 189–191.

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

[41] A. Heryudono, E. Larsson, A. Ramage, L. Von Sydow, *Preconditioning for radial basis function partition of unity methods*, J. Sci. Comput. **67** (2016), pp. 1089–1109.

[42] Y.C. Hon, R. Schaback, *On unsymmetric collocation by radial basis functions*, Appl. Math. Comput. **119** (2001), pp. 177–186.

[43] H. Hoppe, *Surface Reconstruction from Unorganized Points*, Ph.D. Thesis, University of Washington, Washington, 1994.

[44] H. Hoppe, T. Derose, T. Duchamp, J. Mcdonald, W. Stuetzle, *Surface reconstruction from unorganized points*, in: J.J. Thomas (Ed.), Proceedings of the 19-th annual conference on Computer graphics and interactive techniques, ACM press, New York, vol. 26, 1992, pp. 71–78.

[45] H. Hotelling, *Analysis of a complex of statistical variables into principal components*, J. Educ. Psych. **24** (1993), pp. 417–441.

[46] A. Iske, *Multiresolution Methods in Scattered Data Modelling*, Lecture Notes in Computational Science and Engineering Vol. 37, Springer (2004).

[47] I.T. Jolliffe, *Principal Component Analysis*, Springer-Verlag, New York, 1986.

[48] E.J. Kansa, *Application of Hardy's multiquadric interpolation to hydrodynamics*, in: Proc. 1986 Simul. Conf. **4**, 1986, pp. 111–117.

[49] L. Ling, R. Opfer, R. Schaback, *Results on meshless collocation techniques*, Eng. Anal. Bound. Elem. **30** (2006), pp. 247–253.

[50] M. Mathias, *Über positive Fourier-Integrale*, Math. Z. **16** (1923), pp. 103–125.

[51] D.H. McLain, *Two dimensional interpolation from random data*, Comput. J. **19** (1976), pp. 178–181.

[52] J. Meinguet, *Surface spline interpolation: Basic theory and computational aspects*, in: S.P. Singh et al. (Eds.), Approximation Theory and Spline Functions, Reidel, Dordrecht, 1984, pp. 127–142.

[53] J. Meinguet, *Multivariate interpolation at arbitrary points made simple*, Z. Angew. Math. Phys. **30** (1979), pp. 292–304.

[54] J. Meinguet, *An intrinsic approach to multivariate spline interpolation at arbitrary points*, in: N.B. Sahney (Ed.), Polynomial and Spline Approximations, Reidel, Dordrecht, 1979, pp. 163–190.

[55] J. Meinguet, *Basic mathematical aspects of surface spline interpolation*, in: G. Hammerlin (Ed.), Numerische Integration, Birkhauser, Basel, 1979, pp. 211–220.

[56] J.M. Melenk, I. Babuška, *The partition of unity finite element method: Basic theory and applications*, Comput. Meth. Appl. Mech. Eng. **139** (1996), pp. 289–314.

[57] J. Mercer, *Functions of positive and negative type and their connection with the theory of integral equations*, Phil. Trans. Royal Society **209** (1909), pp. 415–446.

[58] C.D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM (Philadelphia), 2000.

[59] C.A. Micchelli, *Interpolation of scattered data: Distance matrices and conditionally positive definite functions*, Constr. Approx. **2** (1986), pp. 11–22.

[60] B. Morse, T.S. Yoo, P. Rheingans, D.T. Chen, K.R. Subramanian, *Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions*, in: IEEE International Conference on Shape Modeling and Applications 2001, IEEE Press, Los Alamitos, 2001, pp. 89–98.

[61] Y. Ohtake, A. Belyaev, H.P. Seidel, *Sparse surface reconstruction with adaptive partition of unity and radial basis functions*, Graph. Models **68** (2006), pp. 15–24.

[62] Jonas Persson and Lina von Sydow , *Pricing European Multi-asset Options Using a Space-time Adaptive FD-method*, Technical report 2003-059 (2003), Department of Information Technology Uppsala University.

[63] Ulrika Pettersson, Elisabeth Larsson, Gunnar Marcusson and Jonas Persson, *Improved basis function methods for mult-dimensional option pricing*, J. Comput. Appl. Math. **222** (2008), pp. 82–93.

[64] A. Safdari-Vaighani, A. Heryudono, E. Larsson, *A radial basis function partition of unity collocation method for convection-diffusion equations arising in financial applications*, J. Sci. Comput. **64** (2015), pp. 341–367.

[65] R. Schaback, *Convergence of unsymmetric kernel-based meshless collocation methods*, SIAM J. Numer. Anal. **45** (2007), pp. 333–351.

[66] I.J. Schoenberg, *Metric spaces and completely monotone functions*, Ann. Math. **39** (1938), pp. 811–841.

[67] L.L. Schumaker, *Spline Functions - Basic Theory*, Wiley-Interscience, New York 1981.

[68] V. Shcherbakov, E. Larsson, *Radial basis function partition of unity methods for pricing vanilla basket options*, Comput. Math. Appl. **71** (2016), pp. 185–200.

[69] D. Shepard, *A two-dimensional interpolation function for irregularly spaced data*, in: Proceedings of 23-rd National Conference, Brandon/Systems Press, Princeton, 1968, pp. 517–524.

[70] H Stehfest, *Algorithm 368: Numerical inversion of Laplace transforms*, Commun. ACM **13** (1970), pp. 47–49.

[71] J. Stewart, *Positive definite function and generalizations, an historical survey*, Rocky Mountain J. Math. **6** (1976), pp. 409–434.

[72] D.Tavella, C.Randall, *Pricing finantial instuments: the Finite Difference Method*, John Wiey & Son, Ed. New York (2000)

[73] G. Turk, J.F. O'Brien, *Modelling with implicit surfaces that interpolate*, ACM Trans. Graph. **21** (2002), pp. 855–873.

[74] H. Wendland, *Scattered Data Approximation*, Cambridge Monogr. Appl. Comput. Math., vol. 17, Cambridge Univ. Press, Cambridge, 2005.

[75] H. Wendland, *Fast evaluation of radial basis functions: Methods based on partition of unity*, in: C.K. Chui et al. (Eds.), Approximation Theory X: Wavelets, Splines, and Applications, Vanderbilt Univ. Press, Nashville, 2002, pp. 473–483.

[76] W.S.B. Woolhouse, *Explanation of a new method of adjusting mortality tables, with some observations upon Mr. Makeham's modification of Gompertz's theory*, J. Inst. Act. **15** (1870), pp. 389–410.

[77] G. Yao, J. Duo, C.S. Chen, L.H. Shen, *Implicit local radial basis function interpolations based on function values*, Appl. Math. Comput. **265** (2015), pp. 91–102.

[78] Z. Wu, R. Schaback, *Local error estimates for radial basis function interpolation of scattered data*, IMA J. Numer. Anal. **13** (1993), pp. 13–27.

[79] S. Zhu, A.J. Wathen, *Convexity and solvability for compactly supported radial basis functions with different shapes*, J. Sci. Comput. **63** (2015), pp. 862–884.