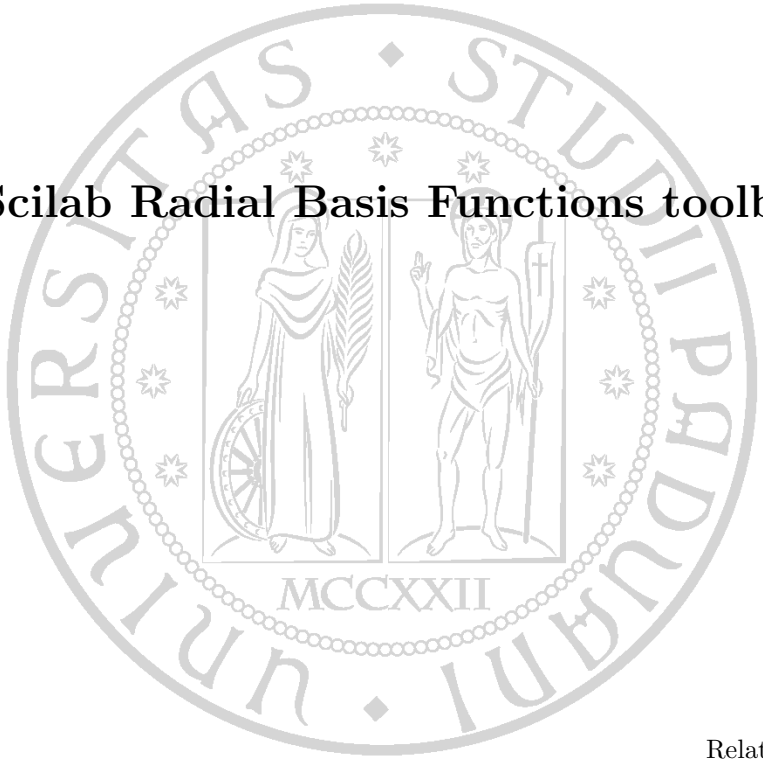
Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA PURA E APPLICATA
Corso di Laurea in Matematica Applicata

Tesi di laurea magistrale

A Scilab Radial Basis Functions toolbox



Candidato:
Anna Bassi
Matricola 625625

Relatore:
Prof. Stefano De Marchi

Correlatore:
Dott.sa Silvia Poles

Anno Accademico 2011–2012

Contents

Introduction

The aim is to develop a toolbox where Radial Basis Functions are used in the scattered data interpolation problem. In this toolbox we construct an approximation function which is a linear combination of shifted RBFs. We have shaped our toolbox to be a powerful instrument to analyze data sets coming both from real life applications and industry. The main advantage of using RBFs is that this class is very large, allowing to model various situations. Nevertheless, there exists a general theory that ensures high tractability of RBFs models. Furthermore, there is emprical evidence that such models give good predictions even with little data. We have chosen Scilab with a double motivation. The first one is that it is an advanced and robust language that guarantees robustness and quick prototyping. In addition, it is an open source software, making our work available for free to any member of the community and industry partners.
This software allows to create an interpolation model based on scattered data from a physical or computer experiment. Here, an experiment is a collection of pairs of input and respective evaluation values. The input is likely to be high dimensional, whereas the measurements have to be scalars.
Computation with high dimensional data is an important issue in many areas of science and engineering. Many traditional numerical methods can either not handle such problems at all, or are limited to very regular situations. Moreover, the independence of meshfree discretizations from a mesh (they are based only on a set of independent points) permits to eliminate the costs of mesh generation, that is still the most time consuming part of any mesh-based numerical simulation.

Chapter 1 contains a description of the Scilab software, in Chapter 2 we give the theory about the RBFs and the RBF interpolation method. Chapter 3 discusses the implementation details and in Chapter 4 we present an example based on real data. Chapter 5 is a reference manual for the developed toolbox.

Chapter 1

Scilab

Scilab is an open source, cross-platform numerical computational package as well as a high-level, numerically oriented programming language. Scilab was written by INRIA, the French National Research Institution, in 1990. The web page for Scilab is www.scilab.org.
Scilab is usually compared to MATLAB® because their syntax and functionalities are very similar, and it is considered its Open Source clone. MATLAB® is a registered trademark of The MathWorks, Inc.
The use of Open Source software offers to companies, that already use commercial equivalent products, the chance of significant savings on purchases and renewals of licenses. In Figure 1.1 an example of return on investment is shown. The high initial cost is due to the migration cost and to the software training cost, while starting from the second year the annual total cost is much lower than the total cost of a commercial competitor software.

Scilab differs from other products in the market (*e.g.* GNU Octave, Maxima, FreeMat) because of its maturity: it has a really advanced plotting system and also includes Xcos, developed with the same idea of MATLAB®'s Simulink, which allows to model and simulate hybrid dynamical systems, such as mechanical, hydraulic, and/or electronic systems. Moreover, Scilab allows for an immediate translation of MATLAB® code.
While the improvement of GNU Octave is left to the users, actually Scilab's

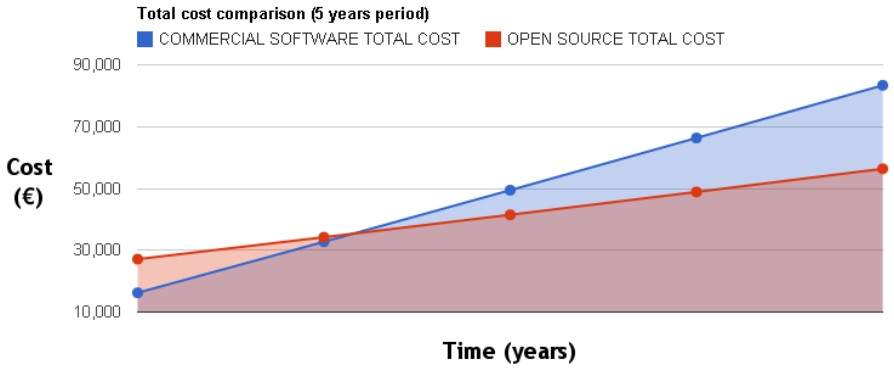


Figure 1.1: Open source software return on investment.

development and maintenance are carefully managed and controlled by the Scilab Consortium, a powerful ecosystem gathering 22 international Consortium members, which includes both industrial and academic members, like PSA Peugeot Citroën and École Polytechnique ParisTech.
Scilab is a European product with six new releases per year and the Scilab Consortium is currently involved in 9 Research and Development projects. Furthermore, an annual user conference is organized, ScilabTec.
For a comparison between Scilab and some of the most used scientific software, as MATLAB®, GNU Octave and FreeMat, we refer to [7].

My personal experience with Scilab
During my studies I took some courses in numerical methods where I learnt programming in MATLAB®. My knowledge was immediately transferable to Scilab and I could start coding with little additional effort.
Nevertheless, one has to be careful while using commands having the same name in both software, since the interpretation of the input could be slightly different.

Chapter 2

Radial Basis Functions Interpolation

2.1 The scattered data interpolation in $\mathbb{R}^s$

We are trying to find a function P_f which is a "good" fit to the given set of data (measurements and locations at which these measurements were obtained) and gives us a rule which allows us to deduce information about the process we are studying also at locations different from those in which we obtained our measurements. We want the function P_f to exactly fit the given measurements at the corresponding locations. If the locations in which the measurements are taken do not lie on a uniform or regular grid, then this approach is called *scattered data interpolation*. To give a precise definition we assume that the measurement locations (or *data sites*) are labelled $\mathbf{x}_j,\ j = 1, \ldots, N$, and the corresponding measurements (or *data values*) are called y_j. We will use $\mathcal{X}$ to denote the set of data sites and assume that $\mathcal{X} \subset \Omega$ for some region Ω in $\mathbb{R}^s$, and restrict the discussion to scalar-valued data, *i.e.*, $y_j \in \mathbb{R}$. We are now ready for a precise formulation of the scattered data interpolation problem.

Problem 2.1.1 (Scattered data interpolation) *Given data* $(\mathbf{x}_j, y_j),\ j = 1, \ldots, N$, *with* $\mathbf{x}_j \in \mathbb{R}^s$ *and* $y_j \in \mathbb{R}$, *find a (continuous) function* P_f *such that* $P_f(\mathbf{x}_j) = y_j,\ j = 1, \ldots, N$.

The fact that we allow $\mathbf{x}_j$ to lie in an arbitrary $s-$dimensional space $\mathbb{R}^s$ means that the formulation of Problem 2.1.1 allows us to cover many different types of applications.
A convenient and common approach to solving the scattered data interpolation problem is to make the assumption that the function P_f is a linear

combination of certain *basis functions* B_k, *i.e.*,

$$P_f(\mathbf{x}) = \sum_{k=1}^{N} c_k B_k(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^s \tag{2.1}$$

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

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

where the entries of the *interpolation matrix* A are given by $A_{jk} = B_k(\mathbf{x}_j),\ j,k = 1,\ldots,N,\ \mathbf{c} = [c_1,\ldots,c_N]^T$, and $\mathbf{y} = [y_1,\ldots,y_N]^T$.
Problem 2.1.1 will be *well posed*, *i.e.*, a solution to the problem will exist and be unique, if and only if the matrix A is non-singular.
In the univariate setting it is well known that one can interpolate to arbitrary data at N distinct data sites using a polynomial of degree $N-1$. For the multivariate setting, however, there is the following negative result due to Mairhuber and Curtis:

Theorem 2.1.2 (Mairhuber-Curtis) *If $\Omega \subset \mathbb{R}^s$, $s \geq 2$, contains an interior point, then there exist no Haar spaces of continuous functions except for one-dimensional ones.*

In order to understand this theorem we need

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

$$\det(A) \neq 0$$

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

Note that existence of a Haar space guarantees invertibility of the interpolation matrix A, *i.e.*, existence and uniqueness of an interpolant to data specified at $\mathbf{x}_1,\ldots,\mathbf{x}_N$, from the space $\mathcal{B}$. As mentioned above, univariate polynomials of degree $N-1$ form an $N-$dimensional Haar space for data given at $\mathbf{x}_1,\ldots,\mathbf{x}_N$.
The Mairhuber-Curtis Theorem tells us that if we want to have a well-posed multivariate scattered data interpolation problem, we can no longer fix in advance the set of basis functions we plan to use for interpolation of arbitrary scattered data. For example, it is not possible to perform unique interpolation with (multivariate) polynomials of degree N to data given at arbitrary

locations in $\mathbb{R}^2$. Instead, the basis should depend on the data locations. The simplest example of basic function consists of the shifts of the Euclidean distance between the data sites in $\mathbb{R}^s$. In other words, we can construct a univariate piecewise linear spline interpolant of a function f by assuming P_f is of the form

$$P_f(\mathbf{x}) = \sum_{k=1}^{N} c_k \parallel \mathbf{x} - \mathbf{x}_k \parallel_2, \quad \mathbf{x} \in \mathbb{R}^s$$

and then determine the coefficients c_k by solving the linear system

$$\begin{pmatrix} \parallel \mathbf{x}_1 - \mathbf{x}_1 \parallel_2 & \parallel \mathbf{x}_1 - \mathbf{x}_2 \parallel_2 & \dots & \parallel \mathbf{x}_1 - \mathbf{x}_N \parallel_2 \\ \parallel \mathbf{x}_2 - \mathbf{x}_1 \parallel_2 & \parallel \mathbf{x}_2 - \mathbf{x}_2 \parallel_2 & \dots & \parallel \mathbf{x}_2 - \mathbf{x}_N \parallel_2 \\ \vdots & \vdots & \ddots & \vdots \\ \parallel \mathbf{x}_N - \mathbf{x}_1 \parallel_2 & \parallel \mathbf{x}_N - \mathbf{x}_2 \parallel_2 & \dots & \parallel \mathbf{x}_N - \mathbf{x}_N \parallel_2 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_N) \end{pmatrix}$$

Clearly, the basis functions $B_k = \parallel \cdot - \mathbf{x}_k \parallel_2$ are dependent on the data sites as suggested by the Mairhuber-Curtis theorem. The points $\mathbf{x}_k$ to which the basic function $B(\mathbf{x}) = \parallel \mathbf{x} \parallel_2$ is shifted are usually referred to as *centers*. While there may be circumstances that suggest choosing these centers different from the data sites one generally picks the centers to coincide with the data sites. This simplifies the analysis of the method, and is sufficient for many applications. Since the functions B_k are (radially) symmetric about their centers $\mathbf{x}_k$ this constitutes the first example of *radial basis functions*.
The matrix above is an example of a *distance matrix*. It is known that the distance matrix based on the Euclidean distance between a set of distinct points in $\mathbb{R}^s$ is always non-singular (Courant-Fisher), therefore, we can solve the scattered data interpolation problem.
Here we present an essential routine of our toolbox, it is called `DistanceMatrix.sci` and we use it to compute the matrix of pairwise Euclidean distances of two (possibly different) sets of points in $\mathbb{R}^s$. In the code these two sets are denoted by `dsites` and `ctrs`. In our examples both of these sets will coincide with the set $\mathcal{X}$ of data sites.

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

```
//   DM:     MxN matrix whose i,j position contains the Euclidean
//              distance between the i-th data site and j-th center

  function DM = DistanceMatrix(dsites,ctrs)
  [M,s] = size(dsites); [N,sc] = size(ctrs);
  if s ~= sc then
     error('dsites and ctrs must have the same number of columns');
  end
  DM = zeros(M,N);
  // Accumulate sum of squares of coordinate differences
  for d = 1:s
     DM = DM + (ones(1,N).*.dsites(:,d)-ones(M,1).*.ctrs(:,d)').^2;
  end
  DM = sqrt(DM);
  endfunction
```

2.2 Radial (basis) functions

The use of the Euclidean distance matrices to compute a solution to the scattered data interpolation problem is quite natural, but it holds some limitations, as a limited accuracy and limited smoothness. We can maintain the underlying structure presented by the distance matrix approach and address these limitations by composing the distance function with certain "good" univariate functions. Radial functions have the nice property that they are invariant under all Euclidean transformations (*i.e.*, translations, rotations, and reflections). By this we mean that it does not matter whether we first compute the RBF interpolant and then apply an Euclidean transformation, or if we first transform the data and then compute the interpolant. This is an immediate consequence of the fact that Euclidean transformations are characterized by orthogonal transformation matrices and are therefore norm-invariant. Invariance under translation, rotation and reflection is often desirable in applications. Moreover, the application of radial functions to the solution of the scattered data interpolation problem benefits from the fact that the interpolation problem becomes insensitive to the dimension s of the space in which the data sites lie. Instead of having to deal with a multivariate function Φ (whose complexity will increase with increasing space dimension s) we can work with the same univariate function φ for all choices of s. We therefore define

Definition A function $\Phi : \mathbb{R}^s \to \mathbb{R}$ is called *radial* provided there exists a

univariate function $\varphi : [0, \infty) \to \mathbb{R}$ such that

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

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

Such a definition says that for a radial function Φ

$$\| \mathbf{x}_1 \| = \| \mathbf{x}_2 \| \implies \Phi(\mathbf{x}_1) = \Phi(\mathbf{x}_2), \qquad \mathbf{x}_1, \ \mathbf{x}_2 \in \mathbb{R}^s$$

In other words, the value of Φ at any point at a certain fixed distance from the origin (or any othe fixed center point) is constant. Thus, Φ is radially (or spherically) symmetric about its center. Furthermore, the definition shows that the Euclidian distance function we took as example of basic function in the previous subsection is just a special case of a radial basis function. Namely, with $\varphi(r) = r$.
Coming back to the scattered data interpolation problem, we now use a radial basis function expansion to solve it in $\mathbb{R}^s$ by assuming

$$P_f(\mathbf{x}) = \sum_{k=1}^{N} c_k \varphi \left(\| \mathbf{x} - \mathbf{x}_k \|_2 \right), \quad \mathbf{x} \in \mathbb{R}^s.$$

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

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

As the solution of the scattered data interpolation problem hinges entirely on the solution of this system of linear equations, the most important question we have to answer is: for what type of basic functions φ is the system matrix non-singular? We will consider this problem in the next sections of this chapter.

2.3 Positive definite RBFs

Let A the matrix with entries $\varphi(\| \mathbf{x}_j - \mathbf{x}_k \|_2), \ j, k = 1, \dots, N$. Our goal is to solve the linear system

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

choosing φ such that A results non-singular, and consequently obtain a unique solution. While no one has yet succeeded in characterizing the class

of all basic functions φ that generate a non-singular system matrix for any set $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of distinct data sites, the situation is much better if we consider *positive definite matrices*.

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

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

for $\mathbf{c} = [c_1, \ldots, c_N]^T \in \mathbb{R}^N$.
If the quadratic form is zero only for $\mathbf{c} \equiv \mathbf{0}$, then A is called *positive definite*.

An important property of positive definite matrices is that all their eigenvalues are positive, and therefore a positive definite matrix is non-singular (but certainly not vice versa).
If we therefore had basis functions B_k in the expansion (2.1) above which generate a positive definite interpolation matrix, we would always have a well-posed interpolation problem. To this end we introduce the concept of a *positive definite function* from classical analysis.

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

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

for any N pairwise different points $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^s$ and $\mathbf{c} = [c_1, \ldots, c_N]^T \in \mathbb{C}^N$.
The function Φ is called *strictly positive definite on* $\mathbb{R}^s$ if the quadratic form is zero only for $\mathbf{c} \equiv \mathbf{0}$.

The left hand side of the inequality is real thanks to Property 3 that we will give in the following.
Some basic properties of positive definite functions are:

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

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

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

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

3. $\Phi(-\mathbf{x}) = \bar{\Phi(\mathbf{x})}$.

4. Any positive definite function is bounded. In fact
$$|\Phi(\mathbf{x})| \leq \Phi(\mathbf{0}).$$

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

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

The integral characterizations of definite positive functions and their extensions to strictly positive definite and strictly completely/multiply monotone functions are essential to the application of the theory to the scattered data interpolation problem:

Theorem 2.3.1 (Bochner) *A (complex-valued) function $\Phi \in C(\mathbb{R}^s)$ is positive definite in $\mathbb{R}^s$ if and only if it is the Fourier transfom of a finite non-negative Borel measure μ on $\mathbb{R}^s$, i.e.*
$$\Phi(\mathbf{x}) = \hat{\mu}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^s}} \int_{\mathbb{R}^s} e^{-i\mathbf{x}\cdot\mathbf{y}} d\mu(\mathbf{y}), \qquad \mathbf{x} \in \mathbb{R}^s.$$

Proof We will prove only the direction that is important for the application to scattered data interpolation. We assume Φ is the Fourier transform of a finite non-negative Borel measure and show that Φ is positive definite. Thus,
$$\begin{aligned}
\sum_{j=1}^{N}\sum_{k=1}^{N} c_j\bar{c}_k\Phi(\mathbf{x}_j - \mathbf{x}_k) &= \frac{1}{\sqrt{(2\pi)^s}} \sum_{j=1}^{N}\sum_{k=1}^{N} \left[c_j\bar{c}_k \int_{\mathbb{R}^s} e^{-i(\mathbf{x}_j-\mathbf{x}_k)\cdot\mathbf{y}} d\mu(\mathbf{y}) \right] \\
&= \frac{1}{\sqrt{(2\pi)^s}} \int_{\mathbb{R}^s} \left[\sum_{j=1}^{N} c_j e^{-i\mathbf{x}_j\cdot\mathbf{y}} \sum_{k=1}^{N} \bar{c}_k e^{i\mathbf{x}_k\cdot\mathbf{y}} \right] d\mu(\mathbf{y}) \\
&= \frac{1}{\sqrt{(2\pi)^s}} \int_{\mathbb{R}^s} \left| \sum_{j=1}^{N} c_j e^{-i\mathbf{x}_j\cdot\mathbf{y}} \right|^2 d\mu(\mathbf{y}) \\
&\geq 0.
\end{aligned}$$

The last inequality holds because of the conditions imposed on the measure μ. $\square$

In order to accomplish our goal to guarantee a well-posed interpolation problem we have to extend Bochner's theorem to *strictly* positive definite functions and we need the notion of *carrier* of a (non-negative) Borel measure defined on some topological space X, given by

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

A sufficient condition for a function to be strictly positive definite is:

Theorem 2.3.2 *Let μ be a non-negative finite Borel measure on $\mathbb{R}^s$ whose carrier is a set of nonzero Lebesgue measure. Then the Fourier transform of μ is strictly positive definite on $\mathbb{R}^s$.*

A criterion to check whether a given function is strictly positive definite is given in the following

Theorem 2.3.3 *Let Φ be a continuous function in $L_1(\mathbb{R}^s)$. Φ is strictly positive definite if and only if Φ is bounded and its Fourier transform is non-negative and not identically equal to zero.*

This theorem is of fundamental importance because it shows that – if $\Phi \neq 0$ (which implies that then also $\hat{\Phi} \neq 0$) – we need to ensure only that $\hat{\Phi}$ be non-negative in order for Φ to be strictly positive definite.
We now turn our attention to (strictly) positive definite *radial* functions. When we are dealing with radial functions, *i.e.*, $\Phi(\mathbf{x}) = \varphi(\|\mathbf{x}\|)$, then it will be convenient to also refer to the univariate function φ as a *(strictly) positive definite radial function*. An immediate consequence of this notational convention is

Lemma 2.3.4 *If $\Phi = \varphi(\|\cdot\|)$ is (strictly) positive definite and radial on $\mathbb{R}^s$, then Φ is also (strictly) positive definite and radial on $\mathbb{R}^\sigma$ for any $\sigma \leq s$.*

In order to give the next result we define the *Bessel function of the first kind of order p*:

$$J_p(x) = \frac{2(2/x)^p}{\sqrt{(}x)\Gamma(1/2 - x)} \int_1^\infty \frac{\sin(xt)}{(t^2-1)^{p+1/2}} dt$$

Now we can give the integral characterizations of this kind of functions, due to Shoenberg:

Theorem 2.3.5 *A continuous function $\varphi : [0, \infty) \to \mathbb{R}$ is positive definite and radial on $\mathbb{R}^s$ if and only if it is the Bessel transform of a finite non-negative Borel measure μ on $[0, \infty)$, i.e.*

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

where

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

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

Theorem 2.3.6 (Shoenberg) *A continuous function $\varphi : [0, \infty) \to \mathbb{R}$ is strictly positive definite and radial on $\mathbb{R}^s$ for all s if and only if it is of the form*

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

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

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

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

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

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

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

- There are no oscillatory univariate continuous functions that are strictly positive definite and radial on $\mathbb{R}^s$ for all s.
- There are no compactly supported univariate continuous functions that are strictly positive definite and radial on $\mathbb{R}^s$ for all s.

2.3.1 Strictly Positive Definite RBFs of the toolbox

We now present a number of functions that are covered by the theory presented so far. It is possible to include a *shape parameter* ε for all the functions presented below by rescaling $\mathbf{x}$ to $\varepsilon\mathbf{x}$. Our use of the shape parameter as a factor applied directly to $\mathbf{x}$ has the advantage of providing a unified treatment in which a decrease of the shape parameter always has the effect of producing "flat" basis functions, while increasing ε leads to more peaked (or

localized) basis functions.
The first strictly positive definite function we consider is well-represented in many branches of mathematics: the *Gaussian* function. Because of its crucial role we are going to show and examine its Fourier transform, while we will just give the formulation and a way to construct the others. For the sake of simplicity also the shape parameter will appear only in the first function.

1. **Gaussians**
   We can now show that the *Gaussian*

   $$\Phi(\mathbf{x}) = e^{-\varepsilon^2\|\mathbf{x}\|^2}, \qquad \varepsilon > 0,$$

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

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

   and this is positive independent of the space dimension s. In particular, for $\varepsilon = \frac{1}{\sqrt{2}}$ we have $\hat{\Phi} = \Phi$.
   Another argument to show that Gaussians are strictly positive definite and radial on $\mathbb{R}^s$ for any s that avoids dealing with Fourier transforms will become available later. It will make use of completely monotone functions.
   Recall that Property 1. shows that any finite non-negative linear combination of (strictly) positive definite functions is again (strictly) positive definite. Now, the Shoenberg characterization of this kind of functions states that *all* such functions are given as infinite linear combinations of Gaussians. Therefore, the Gaussians can be viewed as the fundamental member of the family of functions that are strictly positive definite and radial on $\mathbb{R}^s$ for all s.
   In the toolbox one can pick this function by inserting the string `gaussian` in the field "rbfunction".
   In Figure 2.1 a Gaussian RBF with shape parameter $\varepsilon = 1$ is shown.

2. **Laguerre-Gaussians**
   In order to obtain a generalization of Gaussians we start with the *generalized Laguerre polynomials* $L_n^{s/2}$ *of degree* n *and order* $s/2$ defined by

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

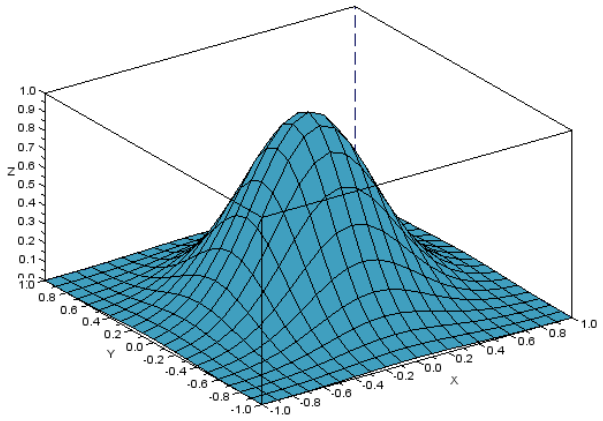


Figure 2.1: Gaussian RBF with shape parameter $\varepsilon = 1$.

We then define the *Laguerre-Gaussians*

$$\Phi(\mathbf{x}) = e^{-\|\mathbf{x}\|^2} L_n^{s/2}(\|\mathbf{x}\|^2).$$

Since they are oscillatory functions we know that they can not be strictly positive definite and radial on $\mathbb{R}^s$ for all s. In fact, in the toolbox the functions `linearLG` (implemented with $n = 1$ and $s = 2$) and `quadraticLG` (implemented with $n = 2$ and $s = 2$) will work only in dimension 2.

3. **Matérn Functions**
The class of *Matérn functions* is quite common in the statistic literature:

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

where K_ν is the *modified Bessel function of the second kind of order* ν. This kind of functions are strictly positive definite on $\mathbb{R}^s$ for all $s < 2\beta$. In the toolbox one can find the functions

- `BasicMatern`: $\beta = \frac{s+1}{2}$, not differentiable at the origin
- `LinearMatern`: $\beta = \frac{s+3}{2}$, C^2 smooth
- `QuadraticMatern`: $\beta = \frac{s+5}{2}$, C^4 smooth
- `CubicMatern`: $\beta = \frac{s+7}{2}$, C^6 smooth

4. **Generalized Inverse Multiquadrics**
   The Henkel inversion theorem allows us to reverse the roles of Φ and $\hat{\Phi}$ of the Matérn functions, generating the so-called *generalized inverse multiquadrics*

$$\Phi(\mathbf{x}) = (1 + \|\mathbf{x}\|^2)^{-\beta}, \qquad \beta > \frac{s}{2},$$

   which are strictly positive definite on $\mathbb{R}^s$ for all $s < 2\beta$.
   In the toolbox it is possible to use the functions

   - `generalizedIMQ`: the "original" inverse multiquadric introduced by Hardy and corresponds to the value $\beta = 1/2$
   - `IQ`: the special choice $\beta = 1$ was referred to as *inverse quadratic* in various papers of Fornberg and co-workers
   - `IMQ`: the inverse multiquadric with $\beta = 2$

2.3.2 Completely Monotone and Multiply Monotone Functions

Fourier transforms are not always easy to compute, thus we present two alternative easier criteria that allow us to decide whether a function is strictly positive definite and radial on $\mathbb{R}^s$ (one for the case of all s and one for only limited choices of s). We begin with the former case. To this end we now introduce a class of functions that is very closely related to positive definite radial functions and leads to a simple characterization of such functions.

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

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

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

In order to see how such functions are related to positive definite radial functions we require the following integral characterization.

Theorem 2.3.7 (Hausdorff-Bernstein-Widder) *A function $\varphi : [0, \infty) \to \mathbb{R}$ is completely monotone on $[0, \infty)$ if and only if it is the Laplace transform of a finite non-negative Borel measure μ on $[0, \infty)$, i.e., φ is of the form*

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

Theorem 2.3.7 shows that, in the spirit of our earlier remarks, the function $\varphi(r) = e^{-\varepsilon r}$ can be viewed as the fundamental completely monotone function.
The following connection between positive definite radial functions and completely monotone function was first pointed out by Shoenberg.

Theorem 2.3.8 *A function φ is completely monotone on $[0, \infty)$ if and only if $\Phi = \varphi(\|\cdot\|^2)$ is positive definite and radial on $\mathbb{R}^s$ for all s.*

Note that the function Φ is now defined via the *square* of the norm.
Also the following *interpolation theorem* originates in the work of Shoenberg and provides a very simple test for verifying the well-posedness of many scattered data interpolation problems.

Theorem 2.3.9 *A function $\varphi : [0, \infty) \to \mathbb{R}$ is completely monotone but not constant if and only if $\varphi(\|\cdot\|^2)$ is strictly positive definite and radial on $\mathbb{R}^s$ for all s.*

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

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

Since convexity of φ means that $\varphi\left(\frac{r_1+r_2}{2}\right) \leq \frac{\varphi(r_1)+\varphi(r_2)}{2}$ or simply $\varphi''(r) \geq 0$ if φ'' exists, a multiply monotone function is in essence just a completely monotone function whose monotonicity is "truncated".
We need an integral representation of this class of function to make the connection to strictly positive definite radial functions.

Theorem 2.3.10 (Williamson) *A continuous function $\varphi : [0, \infty) \to \mathbb{R}$ is $k-$times monotone on $[0, \infty)$ if and only if it is of the form*

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

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

In the RBF literature the following interpolation theorem was stated by Micchelli and refined by Buhmann.

Theorem 2.3.11 (Micchelli) *Let $k = \lfloor s/2 \rfloor + 2$ be a positive integer. If $\varphi : [0, \infty) \to \mathbb{R}$, $\varphi \in C[0, \infty)$, is $k-$times monotone on $(0, \infty)$ but not constant, then φ is strictly positive definite and radial on $\mathbb{R}^s$ for any s such that $\lfloor s/2 \rfloor \leq k - 2$.*

Note that in this case the square of the norm is missing.

2.4 Conditionally Positive Definite Functions

We have not yet the whole picture of radial functions for which the scattered data interpolation problem has a unique solution. There exists a Fourier transform characterization (and the relative easier theory based on monotone functions) also for another class of functions that are not strictly positive definite, namely the *conditionally positive definite functions*.

2.4.1 Scattered data interpolation with polynomial precision

It is not an easy matter to use polynomials to perform multivariate scattered data interpolation. Only if the data sites are in certain special locations we can guarantee well-posedness of multivariate polynomial interpolation.
The following definition guarantees a unique solution for interpolation to give data at a subset of cardinality $M = \binom{m+s}{m}$ of the points $\mathbf{x}_1, \ldots \mathbf{x}_N$, by a polynomial of degree m. Here M is the dimension of the linear space Π_m^s of polynomials of total degree less than or equal to m in s variables.

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

For polynomial interpolation at N distinct data sites in $\mathbb{R}^s$ to be a well-posed problem, the polynomial degree needs to be chosen accordingly, *i.e.*, we need $M = N$, and the data sites need to form a $m-$unisolvent set. This is rather restrictive, however, even though we will be interested in interpolating N pieces of data, the polynomial degree will be small (usually $m = 1, 2, 3$) and the restrictions imposed on the locations of the data sites by the unisolvency conditions will be rather mild.
We now want to modify the assumption on the form of the solution to the scattered data interpolation Problem 2.1.1 by adding certain polynomials to

the expansion, *i.e.*, P_f is now assumed to be of the form

$$P_f(\mathbf{x}) = \sum_{k=1}^{N} c_k \varphi(\|\mathbf{x} - \mathbf{x}_k\|) + \sum_{l=1}^{M} d_l p_l(\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^s \tag{2.2}$$

where $p_1, \ldots, p_M$ form a basis for the $M = \binom{m-1+s}{m-1}-$dimensional linear space Π^s_{m-1} of polynomials of total degree less than or equal to $m-1$ in s variables. Since enforcing the interpolation conditions $P_f(\mathbf{x}_j) = f(\mathbf{x}_j),\ j = 1, \ldots, N$ leads to a system of N linear equations in the $N + M$ unknowns c_k and d_l one usually adds the M conditions

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

to ensure a unique solution.
If the data come from a polynomial of total degree less than or equal to $m-1$, then they are fitted exactly by the expansion 2.2.
In general, solving the interpolation problem based on the extended expansion now amounts to solving a system of linear equations of the form

$$\begin{pmatrix} A & P \\ P^T & O \end{pmatrix} \begin{pmatrix} \mathbf{c} \\ \mathbf{d} \end{pmatrix} = \begin{pmatrix} \mathbf{y} \\ \mathbf{O} \end{pmatrix} \tag{2.3}$$

where the pieces are given by $A_{jk} = \varphi(\|\mathbf{x}_j - \mathbf{x}_k\|),\ j, k = 1, \ldots, N,\ P_{jl} = p_l(\mathbf{x}_j),\ j, = 1, \ldots, N,\ l = 1, \ldots, M,\ \mathbf{c} = [c_1, \ldots, c_N]^T,\ \mathbf{d} = [d_1, \ldots, d_M]^T,\ \mathbf{y} = [y_1, \ldots, y_N]^T, \mathbf{O}$ is a zero vector of length M and O is an $M \times M$ zero matrix. We now need to investigate whether the augmented system matrix in 2.3 is non-singular. The special case $m = 1$ (in any space dimension s) is associated to the reproduction of constants and is covered by standard results from linear algebra, so we discuss it first.

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

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

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

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

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

Note that in order to have a symmetric distance matrix we need to take as centers of the RBFs the same data sites.

Theorem 2.4.1 *Let A be a real symmetric $N \times N$ matrix conditionally positive definite of order one and let $P = [1, \ldots, 1]^T$ be a $N \times 1$ vector. Then the system of linear equation 2.3 is uniquely solvable.*

Proof Assume $[\mathbf{c}, d]^T$ is a solution of the homogeneous linear system,*i.e.*, with $\mathbf{y} = \mathbf{O}$. We show that $[\mathbf{c}, d]^T = \mathbf{O}^T$ is the only possible solution.
By multiplying the top block of the (homogeneous) linear system by $\mathbf{c}^T$ we get

$$\mathbf{c}^T A\mathbf{c} + \mathbf{c}^T Pd = \mathbf{O}.$$

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

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

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

$$A\mathbf{c} + Pd = \mathbf{O},$$

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

Since any strictly positive definite radial function give rise to positive definite matrices, and since positive definite matrices are also conditionally positive definite of order one, the last theorem establishes the non-singularity of the augmented RBF interpolation matrix for constant reproduction.
We now will investigate the case $m > 1$.

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

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

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

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

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

A really similar definition exists in the complex field, thus we can make two useful observations:

- A function that is (strictly) conditionally positive definite of order m on $\mathbb{R}^s$ is also (strictly) conditionally positive definite of any higher order.
- A (strictly) positive definite function is always (strictly) conditionally positive definite of any order.

We can now generalize the interpolation Theorem 2.4.2 to the case of general polynomial reproduction:

Theorem 2.4.2 *If the real-valued even function Φ is strictly conditionally positive definite of order m on $\mathbb{R}^s$ and the points $\mathbf{x}_1, \dots \mathbf{x}_N$ form an $(m-1)-$unisolvent set, then the system of linear equation 2.3 is uniquely solvable.*

2.4.2 Conditionally Positive Definite RBFs of the toolbox

We now present a number of strictly conditionally positive definite (radial) functions, included some of the best known radial basic functions such as the multiquadric due to Hardy and the thin plate spline due to Duchon.

- **Generalized Multiquadrics**

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

  are strictly conditionally positive definite of order $m = \lceil \beta \rceil$ (and higher). For $\beta < 0$ we are back to the generalized inverse multiquadrics, strictly conditionally positive definite of order $m = 0$, namely strictly positive definite.
  In the toolbox one can find

  - `MQ`: the Hardy's "original" multiquadric, with $\beta = \frac{1}{2}$, strictly conditionally positive definite of order 1.
  - `generalizedMQ2`: with $\beta = \frac{3}{2}$, strictly conditionally positive definite of order 2.
  - `generalizedMQ3`: with $\beta = \frac{5}{2}$, strictly conditionally positive definite of order 3.

As in our earlier discussion we can scale the basis functions with a shape parameter ε by replacing $\|\mathbf{x}\|$ by $|\varepsilon|\|\mathbf{x}\|$. This does not affect the well-posedness of the interpolation problem.

- **Radial Powers**

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

are strictly conditionally positive definite of order $m = \lceil \beta/2 \rceil$ (and higher).
This shows that the basic function $\Phi(\mathbf{x}) = \|\mathbf{x}\|_2$ used for the distance matrix are strictly conditionally positive definite of order 1.
The radial powers present in the toolbox are:

  - `linear`: with $\beta = 1$, strictly conditionally positive definite of order 1.
  - `cubic`: with $\beta = 3$, strictly conditionally positive definite of order 2.
  - `quintic`: with $\beta = 5$, strictly conditionally positive definite of order 3.
  - `septic`: with $\beta = 7$, strictly conditionally positive definite of order 4.

We had to exclude even powers because an even power combined with the square root in the definition of Euclidean norm results in a polynomial and we have already decided that polynomials cannot be used for interpolation at arbitrarily scattered multivariate sites.
Radial powers are not affected by a scaling of their argument. In other words, they are *shape parameter free*.

- **Thin Plate Spline**

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

are strictly conditionally positive definite of order $m = \beta + 1$.
The toolbox contains the functions:

  - `TPS`: the "classical" thin plate spline with $\beta = 1$, strictly conditionally positive definite of order 2.
  - `TPS2`: with $\beta = 2$, strictly conditionally positive definite of order 3.

As it happens with radial powers, the use of a shape parameter ε with thin plate splines is pointless.

2.5 Compactly supported RBFs

In Section 2.3 we said that compactly supported radial functions can be strictly positive definite on $\mathbb{R}^s$ only for a fixed maximal $s-$ value. Therefore we focus our attention on the characterization and construction of functions that are compactly supported, strictly positive definite and radial on $\mathbb{R}^s$ for some fixed s.
According to our earlier work, a function is strictly positive definite and radial on $\mathbb{R}^s$ if its $s-$variate Fourier transform is non-negative. In [8] is shown that the Fourier transform of the radial function $\Phi = \varphi(\|\cdot\|)$ can be written as another radial function:

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

where J_ν is the Bessel function of the first kind of order ν.
In the following we define an integral operator and its inverse differential operator, and show how they facilitate the construction of compactly supported radial functions.

Definition

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

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

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

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

The most important properties of these operators are

1. Both $\mathcal{I}$ and $\mathcal{D}$ preserve compact support, *i.e.*, if φ has compact support, then so do $\mathcal{I}\varphi$ and $\mathcal{D}\varphi$.

2. If $\varphi \in C(\mathbb{R})$ and $t \mapsto t\varphi(t) \in L_1[0,\infty)$, then $\mathcal{D}\mathcal{I}\varphi = \varphi$.

3. If $\varphi \in C^2(\mathbb{R})$ ($\varphi \not\equiv 1$) is even and $\varphi\prime(t) \in L_1[0,\infty)$, then $\mathcal{I}\mathcal{D}\varphi = \varphi$.

4. If $t \mapsto t^{s-1}\varphi(t) \in L_1[0,\infty)$ and $s \geq 3$, then $\mathcal{F}_s(\varphi) = \mathcal{F}_{s-2}(\mathcal{I}\varphi)$.

5. If $\varphi \in C^2(\mathbb{R})$ is even and $t \mapsto t^s\varphi\prime(t) \in L_1[0,\infty)$, then $\mathcal{F}_s(\varphi) = \mathcal{F}_{s+2}(\mathcal{D}\varphi)$.

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

Theorem 2.5.1

1. *Suppose $\varphi \in C(\mathbb{R})$. If $t \mapsto t^{s-1}\varphi(t) \in L_1[0,\infty)$ and $s \geq 3$, then φ is strictly positive definite and radial on $\mathbb{R}^s$ if and only if $\mathcal{I}\varphi$ is strictly positive definite and radial on $\mathbb{R}^{s-2}$.*

2. *If $\varphi \in C^2(\mathbb{R})$ is even and $t \mapsto t^s\varphi\prime(t) \in L_1[0,\infty)$, then φ is strictly positive definite and radial on $\mathbb{R}^s$ if and only if $\mathcal{D}\varphi$ is strictly positive definite and radial on $\mathbb{R}^{s+2}$.*

This allows us to construct new strictly positive definite radial functions from given ones by a "dimension-walk" technique that steps through multivariate Euclidean space in even increments.

2.5.1 Compactly Supported RBFs of the toolbox

The following classes of functions illustrate the above mentioned technique.

1. **Wendland's RBFs**
   Wendland starts with the truncated power function

   $$\varphi_l(r) = (1-r)_+^l$$

   which is strictly positive definite and radial on $\mathbb{R}^s$ for $l \geq \lfloor s/2 \rfloor + 1$, and then he walks through dimensions repeatedly applying the operator $\mathcal{I}$:

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

   These new functions are supported in $[0,1]$ and have a polynomial representation there. More precisely,

Theorem 2.5.2 *The functions $\varphi_{s,k}$ are strictly positive definite and radial on $\mathbb{R}^s$ and are of the form*

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

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

The toolbox is provided with Wendland's functions with $s = 3$, which are strictly positive definite and radial on $\mathbb{R}^s$ for $s \leq 3$, and $k = 10, 1, 2, 3$:

- `Wendland30`: of class C^0.
- `Wendland31`: of class C^2.
- `Wendland32`: of class C^4.
- `Wendland33`: of class C^6.

2. **Wu's RBFs**
Wu constructs his functions starting with the functions

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

which in itself is not positive definite, and then he uses the convolution creating

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

This function is strictly positive definite since its Fourier transform is essentially the square of the Fourier transform of ψ and therefore non-negative. Just like the Wendland functions, this function is a polynomial on its support. In fact, the degree of the polynomial is $4l + 1$, and $\psi_l \in C^{2l}(\mathbb{R})$.
Now a family of strictly positive definite radial functions is constructed by a dimension walk using the $\mathcal{D}$ operator:

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

These functions are strictly positive definite and radial on $\mathbb{R}^s$ for $s \leq 2k+1$, are polynomials of degree $4l-2k+1$ on their support and in $C^(2(l-k)$ in the interior of the support. On the boundary the smoothness increases to C^{2l-k}.
The Wu's RBFs available in the toolbox are:

- `Wu03`: of class C^6, usable in $\mathbb{R}$.
- `Wu13`: of class C^4, usable in $\mathbb{R}^s$ with $s \leq 3$.
- `Wu23`: of class C^2, usable in $\mathbb{R}^s$ with $s \leq 5$.
- `Wu03`: of class C^0, usable in $\mathbb{R}^s$ with $s \leq 7$.

2.6 A simple example

We now give a simple example of RBF interpolation: we employ the *Franke bivariate function*, which is a standard test function for $2D$ scattered data fitting:

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

and we will use a set of 1089 uniformly scattered data sites in the unit square in which we sample our test function f. This will be accomplished here by resorting to the so-called *Halton points*. These are uniformly distributed random points in $(0,1)^s$ generated by a low discrepancy sequence that appears to be random, but covers the space more evenly; this kind of points performs very well in low dimensions. The graph of Franke function over the unit square and an example of Halton points are shown in Figure 2.2.

In order to obtain a better approximation we will add to the Halton data sites the corner points $(0,0)$, $(0,1)$, $(1,0)$ and $(1,1)$. As evaluation points we will use a 40×40 uniform grid.
The approximation will be done with five of the best-known RBFs: Gaussian (GA in the plot), Inverse Quadratic (IQ), Inverse Multiquadrics with $\beta = 2$ (IMQ) and with $\beta = 1$ (MQ), and Thin Plate Spline (TPS).
The box-and-whiskers plot of the absolute approximation errors corresponding to the five functions is shown in Figure 2.3. The three horizontal lines of the boxes are the lower quartile Q1, the median and the upper quartile Q3. In order to understand the graphic we give the notion of interquartile range, which is a measure of statistical dispersion, being equal to the difference between the upper and lower quartiles: IQR=Q3−Q1. The lowest

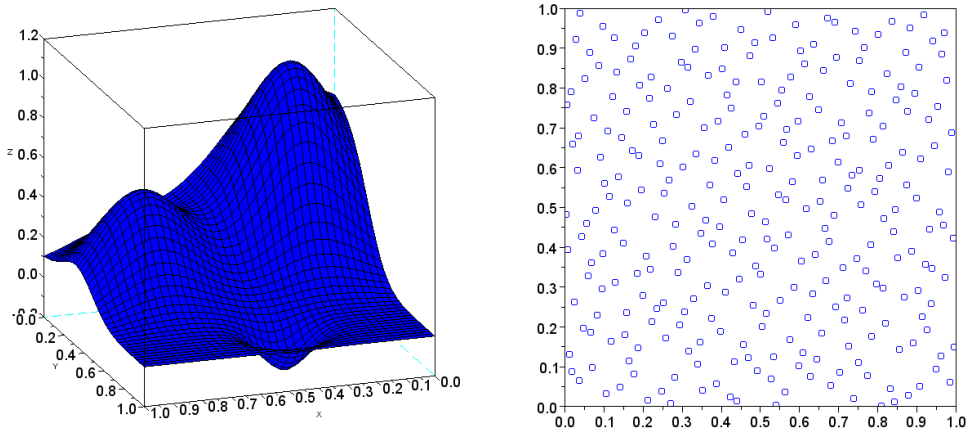


Figure 2.2: Franke bivariate test function and 289 Halton points in the unit square in $\mathbb{R}^2$.

datum of every whisker under the boxes is placed within 1.5IQR of the lower quartile and the highest datum of every whisker on top of the boxes is placed within 1.5IQR of the upper quartile. Any datum not included between the whiskers is plotted as a star. In our case the approximation with Gaussian RBFs results to have the smallest errors on the evaluation points.
The first four RBFs are scaled with the parameters ε that allow the best approximation with that function (in Section 3.4 we will explain how we choose these ε), while TPS is not, since it is shape parameter free.

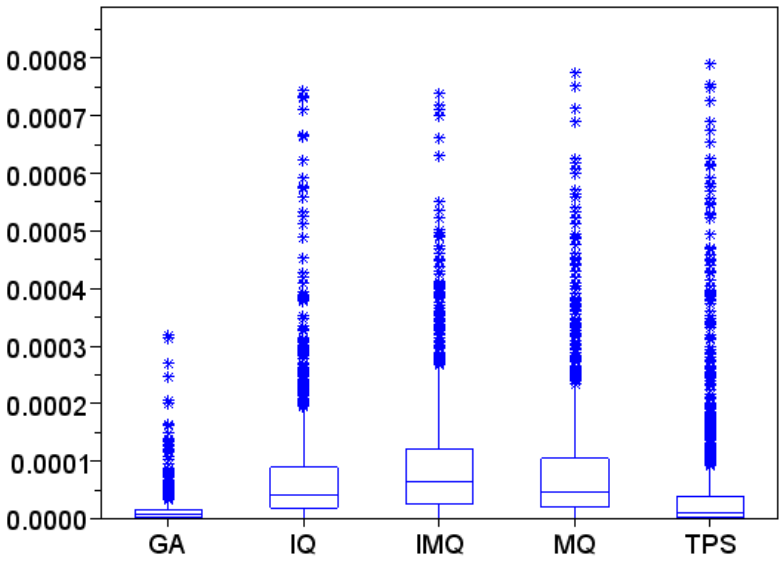


Figure 2.3: Distributions of the absolute approximation errors.

Chapter 3

Implementation details

3.1 Main functions

The toolbox has two major functions:

- `rbfmodel`: this computes the elements of the RBF model, especially the vector of RBF interpolation coefficients $\mathbf{c}$ and the optimal shape parameter ε, calculated through the "Leave One Out" method.

- `rbfapprox`: gives the response of the approximation at the untried sites through the RBF interpolation.

3.2 Repeated points

As it is feasible that an experiment is repeated several times with the same input values, a problem of invertibility of the distance matrix arises, indeed in the presence of repeated points it becomes singular. It is therefore necessary to choose a single point with the respective value given by the measurement and eliminate the remaining.
In the toolbox, we give the user the ability to choose which of the data values will be used in the RBF interpolation, through the optional input parameter `rep_pts_fun` in the function `rbfmodel.sci`. In particular, the eligible options are:

- `fmode`: selects the mode data value, that is the one that most occurs (the default parameter)

- `max`: selects the maximum data value

- `min`: selects the minimum data value
- `fmean`: associates the arithmetic mean of the data values to the repeated point

To minimize the computational cost, the function `DistanceMatrixChecked.sci` sorts the points in a lexicographic order through the use of the Scilab function `gsort`, which performs the sorting by a "quick sort" algorithm for various data types, and checks two by two if they are the same point. In this way we get a $\mathcal{O}(Nlog(N))$ computational cost, instead of $\mathcal{O}(N^2)$.
Now, in case the centers of the RBFs coincide with the data sites (default option), the procedure is over and we can proceed constructing the interpolation matrix. On the other hand, in case they are not the same set of points, we assume that the user will always enter a set of centers with no repetitions. Therefore there is just another problem we should deal with and it arises because the number of centers equals the number of data sites, sure enough, the distance matrix is a square matrix and it could be singular. Hence the function checks if the determinant is equal to zero and, if this condition is satisfied, it perturbs all centers of a 10^{-6} factor.

3.3 The ill-conditioning problem

A standard criterion for measuring the numerical stability of an approximation method is its condition number. In particular, for radial basis function interpolation we need to look at the condition number of the interpolation matrix A with entries $A_{ij} = \Phi(\mathbf{x}_i - \mathbf{x}_j)$. For any matrix A its l_2-*condition number* is given by

$$k_2(A) = \|A\|_2\|A^{-1}\|_2 = \frac{\sigma_{\max}}{\sigma_{\min}},$$

where $\sigma_{\max}$ and $\sigma_{\min}$ are the largest and the smallest singular values of A. If we concentrate on positive definite matrices A, then the condition number of A can also be computed as the ratio

$$\frac{\lambda_{\max}}{\lambda_{\min}}$$

of the largest and the smallest eigenvalues.
Narcowich and Ward established upper bounds on the norm of the inverse of A (lower bounds for $\lambda_{\min}$) in terms of *separation distance* of the data sites

$$q_{\mathcal{X}} = \frac{1}{2}\min_{i\neq j}\|\mathbf{x}_i - \mathbf{x}_j\|_2.$$

We can picture $q_{\mathcal{X}}$ as the radius of the largest ball that can be placed around every point in $\mathcal{X}$ such that two balls do not overlap. Thus, the more the data sites are close together, the more the condition number of the distance matrix increases. The derivation of these bounds is rather technical, and for details we refer to the original papers.
When A is ill-conditioned, *i.e.*, its condition number is large, then most of standard linear system solvers may become unreliable because the solution c_j-values found by these solvers become extremely large in magnitude and a vast amount of numerical cancellation then occurs when P_f is obtained through the combination of these large quantities, with the consequent loss of a significant amount of accuracy.
Moreover, we have seen that the definition of $\Phi(\cdot) = \varphi(\|\cdot\|)$ will sometimes involve a shape or scaling factor ε , for example, the Gaussian basic function, given by $\varphi(\cdot) = e^{-\varepsilon^2\|\cdot\|^2}$, where $\|\cdot\|$ is usually the Euclidean norm. We restrict our attention only to fixed constant $\varepsilon > 0$. It is feasible that the smallest error in RBF interpolation is associated with the choice of a basic function that has a rather flat shape (i.e., ε is small), and therefore the corresponding interpolation matrix is dense and close to singular (due to almost parallel rows or columns of the interpolation matrix).

3.3.1 Scaling

The first operation we apply in our code is to normalize the whole set of input data: in this way we reduce the gap between the data and this allows data on different scales to be compared, by bringing them to a common scale. We perform the scaling in the interval $[0, 1]$. The file `rbfmodel.sci` normalizes the data sites and data values sets and the scaling parameters used here allow the file `rbfapprox.sci` to normalize the evaluation points and to rescale the approximate solution P_f to the order of magnitude of the data values.
We give now a numerical example in order to show the effects of scaling data: in Chapter 4 we will describe a practical situation in which approximating through RBF interpolation could be very useful, and here we are going to use some of the experimental data of that model. At the moment we do not need a description of the data, we just want to find plausible value of the process in four particular locations (two inside and two outside the range where the data sites lie) twice: the first time without normalizing the data, and the second normalizing them.
Besides the data sites and the data value, we also know the values that the process takes in the four evaluation points, so that we can plot the errors that our approximations give. In Figure 3.1 the two approximations and the

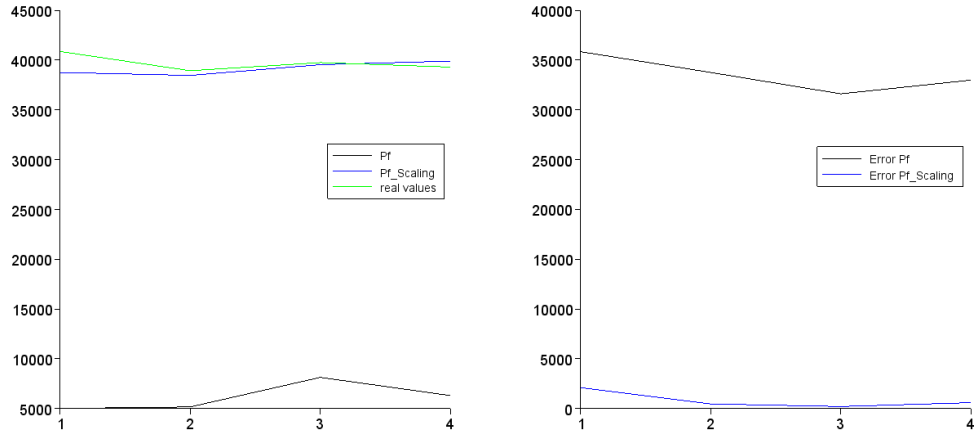


Figure 3.1: On the left, the approximations of the real solution (green) with scaling (blue) and without scaling (black). On the right, the relative errors.

corresponding errors are plotted.

3.3.2 Riley's algorithm

More than fifty years ago, a regularization technique for solving ill-conditioned symmetric positive definite linear systems was developed by J. D. Riley, so we tried to apply it when working with positive definite RBFs and the centers coincide with the data sites, so that the interpolation matrix results to be positive definite and symmetric.
Assuming the goal is to solve the system

$$A\mathbf{x} = \mathbf{b}$$

where A is positive definite, symmetric and ill-conditioned, Riley's algorithm solves a regularized system

$$C\mathbf{y} = \mathbf{b}$$

where

$$C = A + \mu I, \quad \mu > 0.$$

which can be factorized safely with the Cholesky decomposition because if A is symmetric positive definite, so is C.
If all we concerned ourselves with was solving $C\mathbf{y} = \mathbf{b}$ this would be *ridge*

regression or *Tikhonov regularization*. But since $A = C - \mu I$ we can take another step by deriving the identity

$$A^{-1} = C^{-1} \sum_{k=0}^{\infty} \left(\mu C^{-1}\right)^k$$

and therefore, reminding that $\mathbf{y} = C^{-1}\mathbf{b}$, we get the solution to the original system as

$$\begin{aligned}
\mathbf{x} &= A^{-1}\mathbf{b} \\
&= C^{-1} \sum_{k=0}^{\infty} \left(\mu C^{-1}\right)^k \mathbf{b} \\
&= \mathbf{y} \sum_{k=0}^{\infty} \left(\mu C^{-1}\right)^k \\
&= \mathbf{y} + \left(\mu C^{-1}\right) \mathbf{y} + \left(\mu C^{-1}\right)^2 \mathbf{y} + \dots \\
&= \mathbf{y} + \left(\mu C^{-1}\right) \left[\mathbf{y} + \left(\mu C^{-1}\right) \mathbf{y} + \left(\mu C^{-1}\right)^2 \mathbf{y} + \dots\right]
\end{aligned}$$

which gives us a simple iteration method for approximating the solution to $Ax = b$

$$\begin{aligned}
&\mathbf{x}_0 = 0 \\
&\mathbf{x}_{k+1} = \mathbf{y} + \mu C^{-1}\mathbf{x}_k, \qquad k = 0, 1, 2, \dots
\end{aligned} \tag{3.1}$$

This technique allows us to find $\mathbf{x}$ by only performing a stable Cholesky decomposition on C.
We give now another interpretation of Riley's algorithm: let $C = A + \mu I$ as before, then

$$A\mathbf{x} = \mathbf{b} \Leftrightarrow (C - \mu I)\mathbf{x} = \mathbf{b}$$

Now we split A and iterate:

$$\begin{aligned}
&\mathbf{x}_0 = 0 \\
&C\mathbf{x}_{k+1} = \mathbf{b} + \mu\mathbf{x}_k, \qquad k = 0, 1, 2, \dots
\end{aligned} \tag{3.2}$$

which corresponds to (3.1), since $\mathbf{y} = C^{-1}\mathbf{b}$.
In [3] is shown this is equivalent to the *iterative improvement*, a numerical procedure described by [**?**], starting from $\mathbf{x}_0 = 0$:

a. $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$

b. Solve the linear system $C\mathbf{e} = \mathbf{r}_k$

c. $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{e}$

We can easily verify the equivalence

$$\begin{aligned} C\mathbf{x}_{k+1} &\overset{c}{=} C\mathbf{x}_k + C\mathbf{e} \\ &\overset{a,b}{=} C\mathbf{x}_k + \mathbf{b} - A\mathbf{x}_k \\ &\overset{C=A+\mu I}{=} \mathbf{b} + \mu I\mathbf{x}_k \\ &\overset{(3.2)}{=} C\mathbf{x}_{k+1} \end{aligned}$$

The major problem with Tikhonov regularization is the choice of μ. A usual approach could be the *cross validation* or the *maximum likelihood*, but we can do some useful considerations: let the eigenvalues of A be

$$\lambda_{\min} = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n = \lambda_{\max},$$

thus the eigenvalues of μC^{-1} are $0 < \frac{\mu}{\lambda_i+\mu} < 1 \quad \text{for } i = 1, \ldots, n$ and the series

$$\mathbf{x} = \sum_{k=0}^{\infty} \left(\mu C^{-1}\right)^k \mathbf{y}$$

converges. So for $\mu \ll \lambda_{\min}$ we have fast convergence. Furthermore the matrix C is better conditioned than A since

$$k(C) = \frac{\lambda_{\max} + \mu}{\lambda_{\min} + \mu} \ll \frac{\lambda_{\max}}{\lambda_{\min}} = k(A)$$

provided $\mu > \lambda_{\min}$. In summary μ needs to be

- large enough to improve conditioning,
- small enough to provide fast convergence.

Choosing the regularization parameter μ to maximize stability but minimize the summation length is not a straightforward procedure, so it is recommended to use $\mu \approx \lambda_{\min}$. In his paper Riley gives the practical suggestion to choose μ small, precisely

$$\mu \approx 10^{-p+\alpha}$$

where p is the desired precision and $\alpha = 2$ or 3. In our toolbox we fixed μ to the value 10^{-7}, but the user can modify this value in the function `Riley.sci`. A spontaneous question arises: which of the two described forms of the algorithm is more efficient? In order to answer this question we applied both of them in the gaussian RBF interpolation of a set of 1089 Halton data sites evaluated with the Franke bivariate test function. As evaluation points we used a 40×40 uniform grid. We verified that using the same parameters

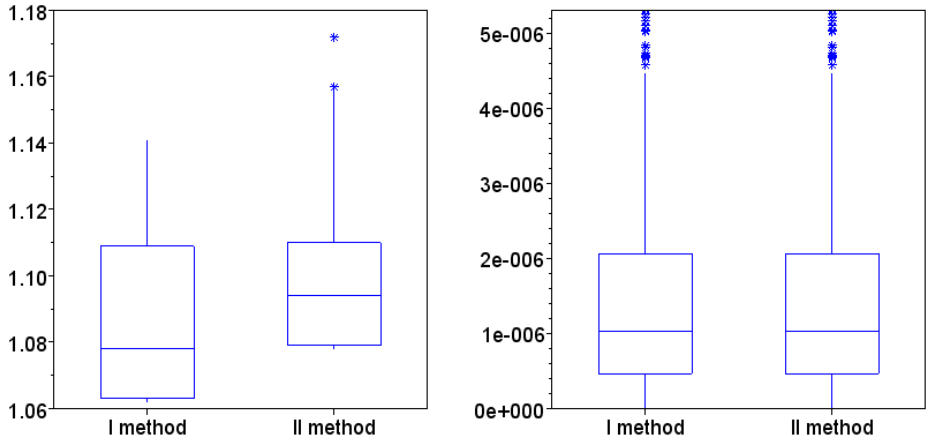


Figure 3.2: A comparison between the two proposed algorithms. On the left the distributions of the times spent, on the right the distributions of the errors.

ε and μ the two algorithms spend almost the same time; in Figure 3.2 the boxplots on the left are related to the distributions of the times employed by the two algorithms (we run them 50 times), while on the right are plotted the boxplots related to the distributions of the absolute approximation errors of the two approximations of the Franke function, the first obtained using the first form and the second obtained using the *iterative improvement*: from the comparison we cannot gather if one of the two methods is better.

In any case we prefer the first form, because, on equal terms, the *iterative improvement* contains subtractions, which could cause the loss of significant digits.

We now make some numerical tests in order to decide when it makes sense to compute the approximation using the Riley's algorithm: the test function will be still the Franke function on $[0,1]^2$, but we will variate the number and the distribution of the data sites. In particular we consider four cases: 289 points on a uniform grid (Figure 3.3), 289 Halton points (Figure 3.4), 1089 points on a uniform grid (Figure 3.5) and 1089 Halton points (Figure 3.6). For every case we give the plots of two absolute approximation errors, on the left the one related to the "classical" approximation and, on the right, the one related to the approximation using Riley's algorithm. In the last two figures we can notice that the errors using the Riley's algorithm decrease of one order of magnitude, thus we choose to activate the Riley's algorithm only if the data sites are more than 1000.

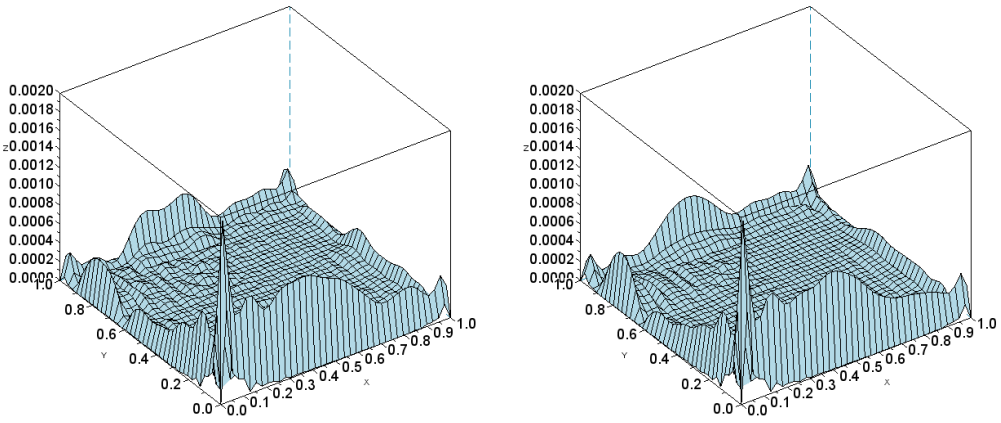


Figure 3.3: Absolute approximation errors without and with Riley's algorithm on a uniform grid 17×17 (289 points).

3.3.3 Benchmark test on the ill-conditioning problem

From what we have said introducing the ill-conditioning problem we can evince that if we want to improve the accuracy of the interpolant adding interpolation points the ill-conditioning increases due to the decrease in the separation distance. In our case we can also consider the situation in which one needs to know the behaviour of the process he is studying in a really small neighbourhood and thus has to use a set of really close together data sites. In this subsection we show with a numerical experiment that even if the condition number is indeed high, reducing the separation distance between the data sites does not compromise the robustness of the RBF interpolation method we have constructed.
We created three sets of points in $\mathbb{R}^2$ and every set consists of 60 points in $[0,1] \times [0,1]$, for the sake of simplicity. 20 of them are pseudo-random, while 40 are created with the help of normal distributions: 20 points are distributed as $\mathcal{N}(0.2, \sigma^2) \times \mathcal{N}(0.2, \sigma^2)$ and the remaining 20 as $\mathcal{N}(0.8, \sigma^2) \times \mathcal{N}(0.8, \sigma^2)$. The three sets differ because of three different values of σ^2: 10^{-2}, 10^{-3} and 10^{-5}. Figure 3.7 shows the first set of data sites we consider.

Remark To give an idea of how the points of the two last sets are concen-

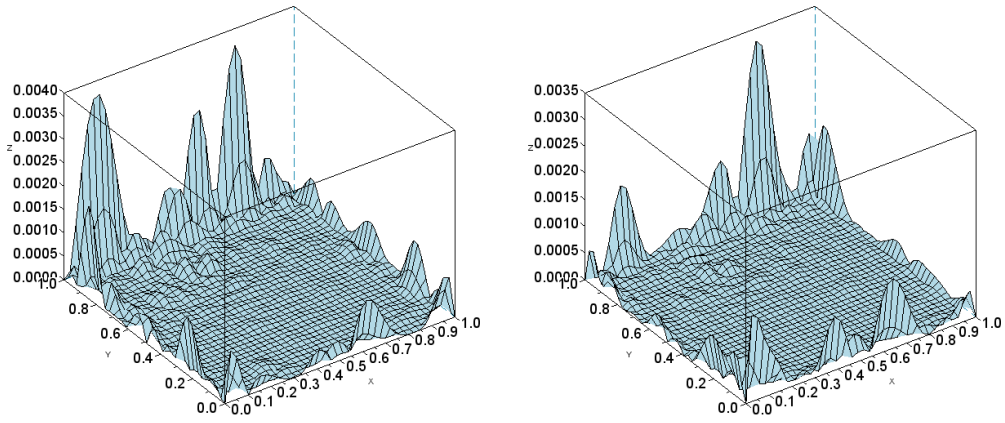


Figure 3.4: Absolute approximation errors without and with Riley's algorithm on 289 Halton points.

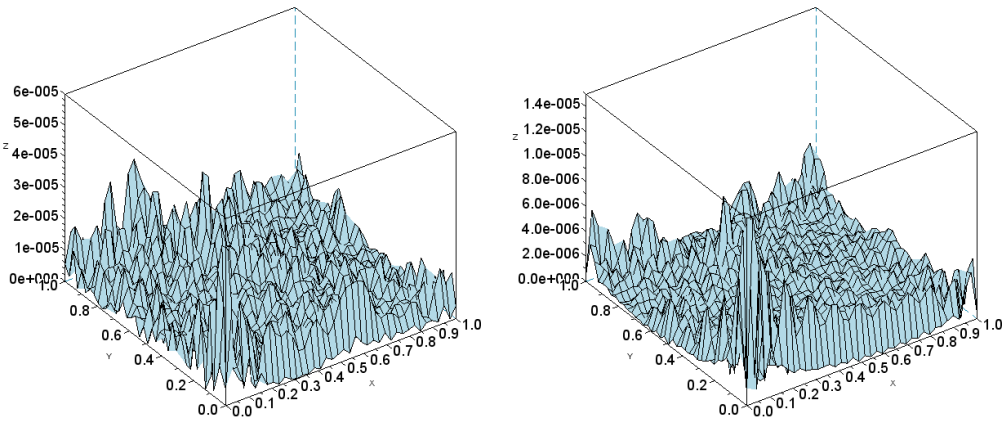


Figure 3.5: Absolute approximation errors without and with Riley's algorithm on a uniform grid 33×33 (1089 points).

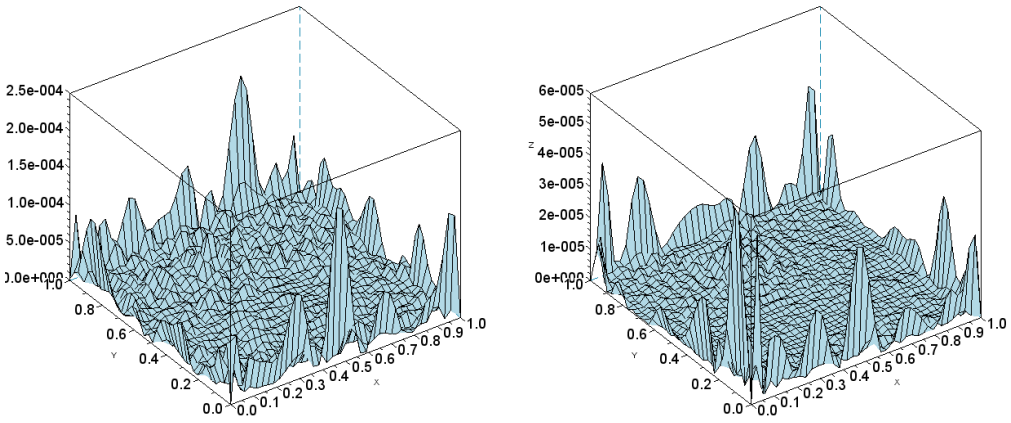


Figure 3.6: Absolute approximation errors without and with Riley's algorithm on 1089 Halton points.

trated around the average points, we point out that if

$$x \sim \mathcal{N}(\mu_x, \sigma_x^2) \quad \text{and} \quad y \sim \mathcal{N}(\mu_y, \sigma_y^2),$$

the probability that a point (x, y) lies outside $\mathcal{B}((\mu_x, \mu_y), R)$ is given by

$$\begin{aligned} P(\|(x,y) - (\mu_x, \mu_y)\| \geq R) &= P(\|(x,y) - (\mu_x, \mu_y)\|^2 \geq R^2) \\ &= P((x-\mu_x)^2 + (y-\mu_y)^2 \geq R^2) \\ &\leq \frac{E((x-\mu_x)^2 + (y-\mu_y)^2)}{R^2}. \end{aligned}$$

Now, by linearity of expectation and definition of σ_x^2 and σ_y^2

$$\begin{aligned} \frac{E((x-\mu_x)^2 + (y-\mu_y)^2)}{R^2} &= \frac{E((x-\mu_x)^2) + E((y-\mu_y)^2)}{R^2} \\ &= \frac{\sigma_x^2 + \sigma_y^2}{R^2}. \end{aligned}$$

Thus, since in our case $\sigma_x^2 = \sigma_y^2 = \sigma^2$, we obtain

$$P(\|(x,y) - (\mu_x, \mu_y)\| \geq R) \leq \frac{2\sigma^2}{R^2}.$$

Coming back to our example, the data values are generated by the function $f = x_1(\sin x_2)$, which we want to approximate through the RBF interpolation

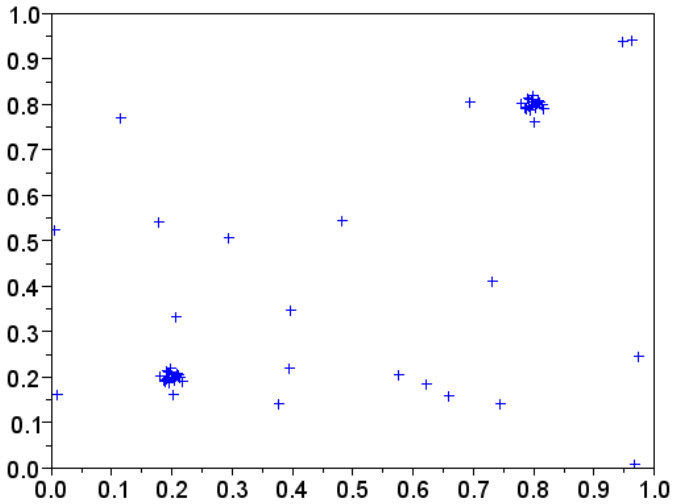


Figure 3.7: First set of data sites ($\sigma^2 = 10^{-2}$)

method. We choose Gaussian RBFs, centers coinciding with data sites, and a 20×20 uniform grid as evaluation points. Table 3.1 shows how a strong decrease of the separation distance $q_{\mathcal{X}}$ does not affect the robustness of the method, in fact in spite of their high condition numbers $k_2(A)$, the three sets of points give very similar approximations, though the separation distance of the third is much smaller.

	$q_{\mathcal{X}}$	$k_2(A)$	RMS Error	Maximum Error
$\sigma^2 = 10^{-2}$	$4.83e-04$	$9.72e+18$	$2.11e-02$	$5.53e-02$
$\sigma^2 = 10^{-3}$	$4.83e-05$	$4.79e+18$	$2.13e-02$	$5.54e-02$
$\sigma^2 = 10^{-5}$	$4.83e-07$	$3.06e+20$	$2.14e-02$	$5.51e-02$

Table 3.1: Benchmark test on the ill-conditioning problem results.

3.4 Choice of the shape parameter

In this section we consider the choice of the shape parameter for a fixed data set. This is probably the situation that will arise most frequently in practical situations. In other words, we assume we are given a set of data $(\mathbf{x}_j, f_j),\ j = 1, \ldots, N$, with data sites $\mathbf{x}_j \in \mathbb{R}^s$ and function values $f_j = f(\mathbf{x}_j) \in \mathbb{R}$. Our goal is to use a RBF interpolant

$$P_f(\mathbf{x}) = \sum_{j=1}^{N} c_j \varphi(\|\mathbf{x} - \mathbf{x}_j\|)$$

to match these data exactly, *i.e.*, to satisfy $P_f(x_i) = f(\mathbf{x}_i),\ i = 1, \ldots, N$. The two most important questions now seem to be:

- Which basic function φ should we use?
- How should we scale the basis functions $\varphi_j = \varphi(\|\cdot - \mathbf{x}_j\|)$?

About the first issue, if we know that the data come from a very smooth function, then application of one of the smoother basic functions is called for. Otherwise, there is not much to be gained from doing so. In fact, these functions may add too much smoothness to the interpolant.
But we want to focus on the second question: we will assume throughout that a (fixed) basic function has been chosen, and that we will use only one value to scale all basis functions uniformly.
The strategy we chose to implement is a *cross validation* approach. In [Rippa (1999)] an algorithm is described that corresponds to a variant of cross validation known as *"leave-one-out"cross validation*. In this algorithm an "optimal"value of ε is selected by minimizing the (least square) error for a fit to the data based on an interpolant for which one of the centers was "left out". A good feature of this method is that the dependence of the error on the data function is taken into account. Specifically, if $P_f^{[k]}$ is the RBF interpolant to the data $\{f_1, \ldots, f_{k-1}, f_{k+1}, \ldots, f_N\}$, *i.e.*,

$$P_f^{[k]}(\mathbf{x}) = \sum_{\substack{j=1 \\ j \neq k}}^{N} c_j^{[k]} \varphi(\varepsilon \|\mathbf{x} - \mathbf{x}_j\|)$$

such that

$$P_f^{[k]}(\mathbf{x}_i) = f_i, \qquad i = 1, \ldots, k-1, k+1, \ldots, N,$$

and if E_k is the error

$$E_k = f_k - P_f^{[k]}(\mathbf{x}_k)$$

at the one point $\mathbf{x}_k$ not used to determine the interpolant, then the quality of the fit is determined by the norm of the vector of errors $E = [E_1, \ldots, E_N]^T$ obtained by removing in turn one of the data points and comparing the resulting fit with the (known) value at the removed point. We will use the maximum norm.
By adding a loop over ε we can compare the error norms for different values of the shape parameter, and choose that value of ε that yields the minimal error norm as the optimal one.
While a naive implementation of the leave-one-out algorithm is rather expensive (on the order of N^4 operations), Rippa showed that the computation of the error components can be simplified to a single formula

$$E_k = \frac{c_k}{A^{-1}_{kk}} \tag{3.3}$$

where c_k id the kth coefficient of the interpolant P_f based on the *full* data set, and A^{-1}_{kk} is the kth diagonal element of the inverse of the corresponding interpolation matrix. Since both c_k and A^{-1} need to be computed only once for each value of ε this results in $\mathcal{O}(N^3)$ computational complexity. We can notice that all entries in the error vector E can be computed in a single statement in Scilab if we vectorize the component formula (3.3).
In order to find a good shape parameter as quickly as possible we implemented the function `rbf_minsearch.sci`, that finds a local minimum of the cost function for ε `CostEpsilon.sci`. Since the minimum will be local, it is opportune to choose a "good " interval in which it will be calculated. An example of the function `CostEpsilon` depending on ε is shown in Figure 3.8 Details on the algorithm of this optimization function are given in Subsection 3.4.1.

3.4.1 Finding a minimum of a function of one variable with `rbf_minsearch`

`rbf_minsearch` is a file function inspired by the MATLAB®'s function `fminbnd`. As MATLAB® reports the outcome may be just a local minimum and not a global minimum, indeed it is known that the algorithm gives an exact minimizer only if the cost function is unimodal.

Definition A function $f(x)$ is a *unimodal function* if for some value m, it is monotonically increasing (decreasing) for $x \leq m$ and monotonically decreasing (increasing) for $x \geq m$. In that case, the maximum (minimum) value of $f(x)$ is $f(m)$ and there are no other local maxima (minima).

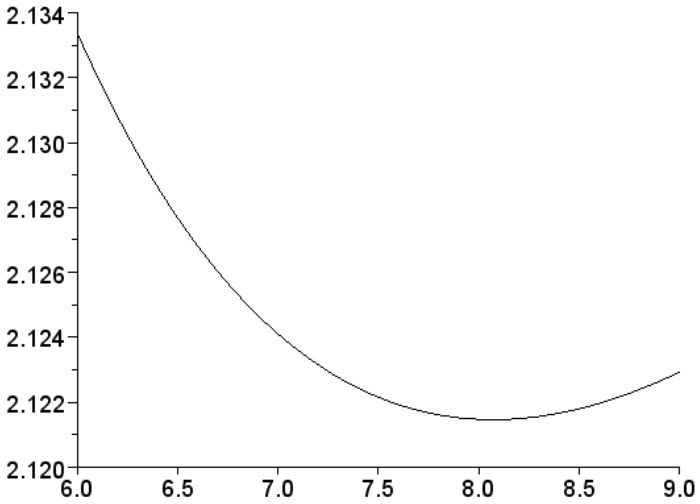


Figure 3.8: $\|E\|_\infty$ of the approximations of a process (described in Chapter 4) calculated by the function `CostEpsilon` in 100 different ε in the interval $(6, 9)$.

The algorithm is based on *golden search* and *parabolic interpolation*.
The *golden search* is a technique for finding the extremum (minimum or maximum) of a unimodal function by successively narrowing the range of values inside which the extremum is known to exist. The technique derives its name from the fact that the algorithm maintains the function values for triples of points whose distances form a golden ratio.
The *parabolic interpolation* starts with three arbitrary real numbers v_1, v_2, v_3. At the general stage one has v_{k-2}, v_{k-1} and v_k. v_{k+1} will be the abscissa of the minimum ordinate of the parabola (with vertical axis) through $(v_i, f(v_i)), i = k-2, k-1, k$.
The algorithm `rbf_minsearch` finds an approximation to the minimum of a function f defined on the interval $[a, b]$. Unless a is very close to b, f is never evaluated at the endpoints a and b, so f needs only to be defined on (a, b), and if the minimum is actually at a or b, then an interior point distant no more than $2tol$ from a or b will be returned, where tol is a tolerance (see equation (3.4) below).
At a typical step of the algorithm there are six significant points: a, b, u, v, w and x, not all distinct. The positions of this points change during the algorithm. Initially (a, b) is the interval on which f is defined, and

$$v = w = x = a + \left(\frac{3-\sqrt{5}}{2}\right)(b-a).$$

The magic number $(3-\sqrt{5})/2$ is rather arbitrarily chosen so that the first step is the same as for a golden section search.
At the start of a cycle the points a, b, u, v, w and x always serve as follows: a local minimum lies in $[a, b]$; of all the points at which f has been evaluated, x is the one with the least value of f, or the point of the most recent evaluation if there is a tie; w is the point of the next lowest value of f; v is the previous value of w; and u is the last point at which f has been evaluated (undefined the first time).
The tolerance is given by

$$tol = \sqrt{eps}|x| + \frac{TolX}{3}, \tag{3.4}$$

where eps is the machine-precision and $TolX$ is fixed to the default value 10^{-8}, but can be modified by the user.
Let $m = \frac{1}{2}(a+b)$ be the midpoint of the interval known to contain the minimum. if $|x-m| \leq 2tol - \frac{1}{2}(b-a)$, *i.e.*, $\max(x-a, b-x) \leq 2tol$, then the procedure terminates with x as the approximate position of the minimum. Otherwise, numbers p and q ($q \geq 0$) are computed so that $x + p/q$

is the turning point of the parabola passing through $(v, f(v))$, $(w, f(w))$ and $(x, f(x))$. If two or more of these points coincide, or if the parabola degenerates, then $q = 0$.
p and q are given by

$$p = (x - v)^2(f(x) - f(v)) - (x - w)^2(f(x) - f(w))$$

$$q = 2((x - v)(f(x) - f(v)) - (x - w)(f(x) - f(w)))$$

Let now e be the value of the correction p/q at the second-last cycle. If $|e| \leq tol, q = 0, x + p/q \notin (a, b)$, or $|p/q| \geq |e|/2$, then a "golden section" step is performed, *i.e.*, the next value of u is

$$u = \begin{cases} \left(\frac{\sqrt{5}-1}{2}\right) x + \left(\frac{3-\sqrt{5}}{2}\right) a, & x \geq m \\ \left(\frac{\sqrt{5}-1}{2}\right) x + \left(\frac{3-\sqrt{5}}{2}\right) b, & x < m \end{cases}$$

Otherwise u is taken as $x + p/q$ (a "parabolic interpolation" step), except that the distances $|u - x|, u - a$ and $b - u$ must be at least tol. Then f is evaluated at the new point u, the points a, b, v, w, and x are updated as necessary, and the cycle is repeated. We see that f is never evaluated at two points closer together than tol.
Typically the algorithm terminates in the following way: $x = b - tol$ (or, symmetrically, $a+tol$) after a parabolic interpolation step has been performed with the condition $|u - x| \geq tol$ enforced. The next parabolic interpolation point lies very close to x and b, so u is forced to be $x-tol$. If $f(u) > f(x)$ then a moves to u, $b - a$ becomes $2tol$, and the termination criterion is satisfied. Note that two consecutive steps of tol are done just before termination. If a golden section search were done whenever the last, rather than second-last, value of $|p/q|$ was tol or less, then termination with two consecutive steps of tol would be prevented, and unnecessary golden section steps would be performed.

3.5 Interpolation with compactly supported RBFs

In this section we illustrate a technique that could be used in order to optimize the RBF interpolation in case of compactly supported functions. The toolbox, in fact, is not yet provided of such optimization codes.
In order to deal with large sets of data efficiently we frequently use *compactly supported* basic functions. For their successful implementation we need to

know which data sites lie in the support of a given basis function (such a query is known as a *range search*). We also may be interested in finding all centers whose support contains a (given) evaluation point (*containment query*). Therefore the main difference to our previous interpolants is that now the interpolation matrix can be made *sparse*, because we do not want to compute the matrix entries for all pairs of points since we know all of the entries for far away points will be zero.
It turns out that it is easier to deal with compact support if we compute the distance matrix corresponding to the $(1 - \varepsilon r)_+$ term, as otherwise those entries of the distance matrix that are zero (since the mutual distance between two identical points is zero) would be lost in the sparse representation of the matrix.
As we know that the interpolation matrix will be sparse, we could write a code to efficiently assemble the matrix. Once we have defined a sparse matrix, Scilab will automatically use state-of-the-art sparse matrix techniques to solve the linear system. An efficient data structure is needed and we suggest $kd-$trees, implemented in a set of MATLAB® files written by Guy Shechter, which can be downloaded by the MATLAB® Central File Exchange.
The code of the file that computes the distance matrix, in this case, contains two similar blocks that will be used depending on whether we have more centers than data sites or vice versa: if there are more data sites than centers, then it will build a $kd-$tree for the data sites and find, for each center, those data sites within the support of the basis function centered at that center, *i.e.*, it constructs the sparse matrix column by column. In the other case it starts with a tree for the centers and builds the matrix row by row.
A $kd-$tree ($k-$dimensional tree) is a space-partitioning data structure for organizing points in $k-$dimensional space. The purpose of $kd-$tree is to decompose a set of N data points in $\mathbb{R}^s$ into a relatively small number of subsets such that each subset contains roughly the same number of data sites. Each node in the tree is defined by a splitting plane that is perpendicular to one of the coordinate axes and passes through one of the data points. Therefore the splitting planes partition the set of points at the median into "left" and "right" (or "top" and "bottom") subsets, each with roughly half the points of the parent node. These children are again partitioned into equal helves, using planes through a different dimension. This partitioning process stops after $\log N$ levels. The computational complexity for building a $kd-$tree from N points in $\mathbb{R}^s$ is $\mathcal{O}(sN \log N)$. Once the tree is built, a range query can be performed in $\mathcal{O}(\log N)$ time. This compares favorably with the $\mathcal{O}(N)$ time it takes to search the "raw" data set.

Chapter 4

A real case approximation

In this chapter we are going to show how useful could be the RBFs interpolation method for companies with a practical example, based on data deriving form measurements taken in different phases of a die casting process.
Die casting is a metal casting process that is characterized by forcing molten metal under high pressure into a mould cavity. After a product is designed, the mould cavity is created using two hardened tool steel dies which have been machined to form the features of the desired part. The high-pressure injection leads to a quick fill of the die, which is required so the entire cavity fills before any part of the casting solidifies. Die casting is widely used for manufacturing a variety of parts, from the smallest component to entire body panels of cars. The solidification of the product is performed through the employment of a cold-chamber machine.
The four stages of the process are:

I *Die preparation*: the dies are prepared by spraying the mould cavity with lubricant. The lubricant has two functions: it helps controlling the temperature of the die and it also assists in the removal of the casting.

II *Filling*: the dies are then closed and molten metal is injected into the dies under high pressure through a first fast injection phase. In order to fill the smallest die cavities a faster second injection phase is performed.

III *Ejection*: once the mould cavity is filled, the pressure is maintained until the casting solidifies. The dies are then opened and the shot is ejected by the ejector pins.

IV *Shakeout*: this action consists in separating the scrap from the shot.

Die casting is a non-expendable mould casting, namely the mould needs not to be reformed after each production cycle. In particular, such a mould is

typically very expensive, therefore it has to be employed for the creation of the largest number of pieces possible.
In this situation the help of an approximation of the process could be really useful: the product engineers may need the development of a visual model and such a model could be used in order to optimize the process with the following aims:

- to maximize the life-time of the mould, because of its cost;
- to minimize the solidification time, in order to maximize the production.

Our first step is to select six experimental parameters that will constitute the data sites in $\mathbb{R}^6$. The die casting process parameters we consider are

1. *holding furnace temperature*, which takes values from 670°C up to 730°C every 10°C;
2. *first phase shot velocity,* which takes values from 0.1m/s up to 1.3m/s every 0.2m/s;
3. *second phase shot velocity*, which takes values from 2.5m/s up to 4.5m/s every 0.5m/s;
4. *shot shift point*, which takes values from 270mm up to 370mm every 5mm;
5. *selected specific pressure*, which takes values from 600bar up to 1000bar every 100bar;
6. *lubrication time*, which takes values from 4s up to 12s every 2s;

while the parameters that will serve as data values in our two approximations are

- *die temperature*, which takes values from 20.3°C up to 20.8°C;
- *solidification time*, which takes values from 10.5s up to 13.7s.

We are going to approximate these two parameters as functions of the holding furnace temperature and the lubrication time and considering, in every data site, the average values of the data values of the other four parameters. This way we can have the 3−dimensional visual models of how the die temperature (Figure 4.1) and the solidification time (Figure 4.2) behave as the holding furnace temperature and the lubrication time change. In particular, in both

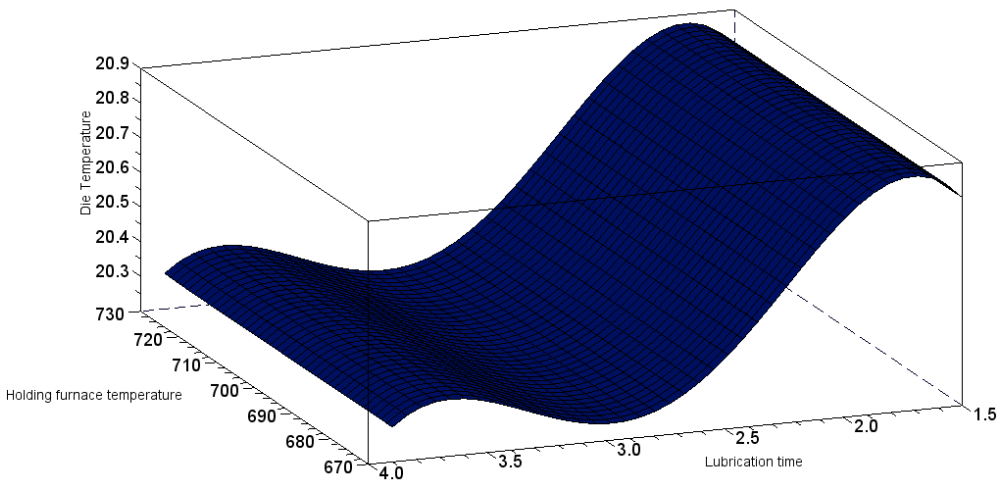


Figure 4.1: The predicted values of the die temperature.

cases we interpolated 150 pairs of data sites and data values, approximating the processes in 40×40 grids of evaluation points. We can see that the die temperature only depends on the lubrication time, while the solidification time increases as both the holding furnace temperature and the lubrication time increase.

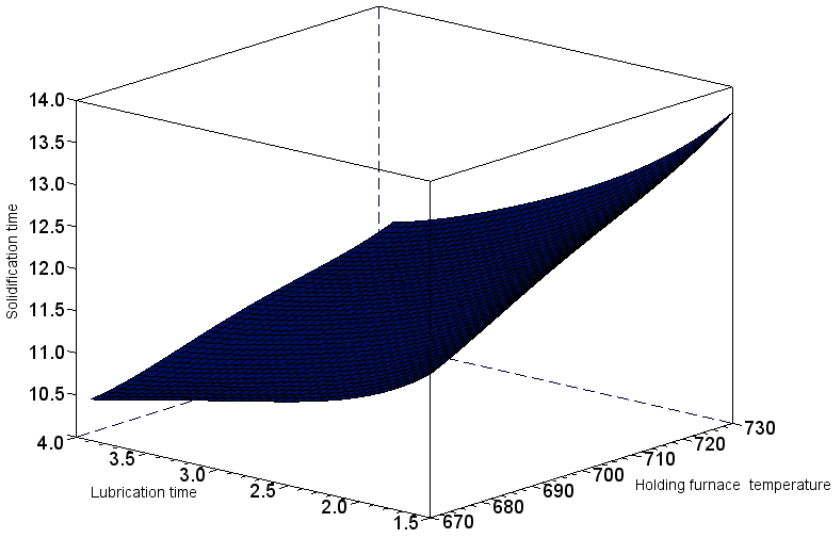


Figure 4.2: The predicted values of the solidification time.

Chapter 5

Reference Manual

This chapter is a presentation of the functions in `RBFtoolbox`. The contents are

5.1 Model Construction	
`rbfmodel`	Finds the RBF model to given sets of data sites and data values and a given RBF

5.2 Evaluate the Model	
`rbfapprox`	Use the RBF model to predict the function at one or more untried sites

5.3 Radial Basis Functions	
`gaussian`	Gaussian
`linearLG`	Linear Laguerre-Gaussian for $\mathbb{R}^2$
`quadraticLG`	Quadratic Laguerre-Gaussian for $\mathbb{R}^2$
`BasicMatern`	Basic Matérn
`LinearMatern`	Linear Matérn
`QuadraticMatern`	Quadratic Matérn
`CubicMatern`	Cubic Matérn
`IQ`	Inverse Quadratic
`IMQ`	Inverse Multiquadric
`generalizedIMQ`	Generalized Inverse Multiquadric
`MQ`	Hardy's Multiquadric
`generalizedMQ2`	Generalized Multiquadric
`generalizedMQ3`	Generalized Multiquadric
`linear`	Linear Radial Power
`cubic`	Cubic Radial Power
`quintic`	Quintic Radial Power
`septic`	Septic Radial Power

`TPS`	Thin Plate Spline
`TPS2`	Second Order Thin Plate Spline
`Wendland30`	Wendland's $\varphi_{3,0}$ for $\mathbb{R}^s,\ s \leq 3$
`Wendland31`	Wendland's $\varphi_{3,1}$ for $\mathbb{R}^s,\ s \leq 3$
`Wendland32`	Wendland's $\varphi_{3,2}$ for $\mathbb{R}^s,\ s \leq 3$
`Wendland33`	Wendland's $\varphi_{3,3}$ for $\mathbb{R}^s,\ s \leq 3$
`Wu03`	Wu's $\psi_{0,3}$ for $\mathbb{R}$
`Wu13`	Wu's $\psi_{1,3}$ for $\mathbb{R}^s,\ s \leq 3$
`Wu23`	Wu's $\psi_{2,3}$ for $\mathbb{R}^s,\ s \leq 5$
`Wu33`	Wu's $\psi_{3,3}$ for $\mathbb{R}^s,\ s \leq 7$

5.4 Repeated Points Functions	
`fmode`	selects the mode data value
`max`	selects the maximum data value
`min`	selects the minimum data value
`fmean`	associates the arithmetic mean of the data values to the repeated point

5.5 Auxiliary Functions	
`DistanceMatrix`	forms the distance matrix of two sets of points in $\mathbb{R}^s$
`Riley`	solves ill-conditioned symmetric positive definite linear systems
`rbf_minsearch`	finds a local minimum of a function in a bounded interval

5.6 Data Files	
`data1`	Example data `dsites` and `dvalues`
`dataFranke`	Example data `dsites` and `dvalues`

5.1 Model Construction

Purpose: Finds the RBF model to given sets of data sites and data values and a given RBF.

Call:

```
cmodel = rbfmodel(dsites, dvalues, rbfunction)
cmodel = rbfmodel(dsites, dvalues, rbfunction, eps)
cmodel = rbfmodel(dsites, dvalues, rbfunction, eps, ctrs)
cmodel = ...
  rbfmodel(dsites, dvalues, rbfunction, eps, ctrs, maxe, mine)
cmodel = ...
  rbfmodel(dsites, dvalues, rbfunction, eps, ctrs, ...
    mine, maxe, rep_pts_fun)
```

Input parameters:

`dsites`	Data sites: an $M \times s$ array.
`dvalues`	Data values: $M \times 1$ array with responses at `dsites`.
`rbfunction`	Handle to a function. See Section 5.3.
`eps`	If present, the shape parameter, otherwise `eps`= 1 is the default value.
`ctrs`	If present, the centers for the RBFs, otherwise they will coincide with `dsites` by default.
`mine`,`maxe`	If present, then lower and upper bounds on the shape parameter.
`rep_pts_fun`	If present, handle to a function. See Section 5.4.

Output:

`cmodel`	RBF model. Struct with the elements	
	`c`	Solution of the interpolation system.
	`cRiley`	If calculated, solution of the interpolation system computed with Riley's algorithm.
	`rbfunction`	Handle to a function. See Section 5.3.
	`eps`	The shape parameter fuond using the leave one out method.
	`ctrs`	If present, the centers for the RBFs.
	`dmin`,`dmax`	Scaling factors for design sites.
	`vmin`,`vmax`	Scaling factors for design responses.

Remark The first step in `rbfmodel` is to normalize the input:

```
for i = 1:size(dsites,2)
```

```
   dmin(i) = min(dsites(:,i)); dmax(i) = max(dsites(:,i));
   dsites(:,i) = (dsites(:,i)-dmin(i))./ (dmax(i)-dmin(i));
end
vmin = min(dvalues); vmax = max(dvalues);
dvalues = (dvalues-vmin)./ (vmax-vmin);
```

The values in `dmin`, `dmax`, `vmin` and `vmax` are returned in `cmodel.dmin`, `cmodel.dmax`, `cmodel.vmin` and `cmodel.vmax`.
This function also checks if the data sites set contains repeated points and selects some of them (and the related data values) through the use of the `rep_pts_fun` function.
If `rbfunction` is strictly conditionally positive definite of order m on $\mathbb{R}^s$, the data sites coincide with the centers and the set of data sites is $m-$unisolvent, then `rbfmodel` modifies the interpolation matrix in order to make the problem uniquely solvable (see Section 2.4) and adding linear precision:

```
[Rd,Cd] = size(dsites);
PM = [ones(Rd,1) dsites];
IM = [IM PM; [PM' zeros(Cd+1,Cd+1)]];
dvalues = [dvalues; zeros(Cd+1,1)];
c = IM\dvalues;
```

In case the distance matrix is symmetric positive definite and the data sites are more than 1000, the funtion `Riley` performs a more stable computation of the solution of the interpolation system.

5.2 Evaluate the Model

Purpose: Use the RBF model to predict the function at one or more untried sites.

Call:

```
Pf = rbfapprox(epoints, cmodel)
[Pf,PfRiley] = rbfapprox(epoints, cmodel)
```

Input parameters:

`epoints`	Evaluation points: trial design sites with s dimensions. For N trial sites `epoints` must be an $N \times s$ matrix with the sites stored rowwise.
`cmodel`	Struct created with `rbfmodel`; see Section 5.1.

Output:

`Pf`	Predicted response at `epoints`.
`PfRiley`	If computed, predicted response using Riley's algorithm at `epoints`.

Remark The computation is performed on normalized trial sites, but the returned results are in the original "units".

5.3 Radial Basis Functions

The toolbox provides 27 functions that implement the RBFs presented in Chapter 2, see the list on page 50. We only present one of them in detail.
Purpose: Get values of the Gaussian function.

Call:

```
rbf = gaussian(e,r)
```

Input parameters:

`e`	Shape parameter.
`r`	Where we want to evaluate the function.

Output:

`rbf`	Value of the Gaussian function in `r`.
`class`	A vector $[class_1, class_2]$ which gives information about the function. $class_1$ can take four values: 1 Strictly positive definite function (Gaussian case). 2 Strictly conditionally positive definite function. 3 Strictly conditionally positive definite and shape parameter free function. 4 Strictly positive definite and compactly supported function. $class_2$ returns the order of a strictly conditionally positive definite function, it takes values between 0 and 4.

Remark The Radial Powers (`linear`, `cubic`, `quintic` and `septic`) and the Thin Plate Splines (`TPS` and `TPS2`) are "shape parameter free", thus the parameter `e` does not affect these functions.

5.4 Repeated Points Functions

We present here only the function `fmode`, that is the default parameter; `fmean`, `min` and `max` work in the same way. In particular, the last two functions are provided by Scilab.
Purpose: Get the mode of a vector, that is the one that most occurs.

Call:

```
[M,k] = fmode(v)
```

Input parameters:

`v` Vector.

Output:

`M` Mode value of `v`.

`k` index of the first position of `M` in `v`.

5.5 Auxiliary Functions

Purpose: Form the distance matrix of two sets of points in $\mathbb{R}^s$. *i.e.* $\texttt{DM}(i,j) = \|\texttt{dsite}_i - \texttt{center}_j\|_2$.

Call:

```
DM = DistanceMatrix(dsites,ctrs)
```

Input parameters:

`dsites` $M \times s$ matrix representing a set of M data sites in $\mathbb{R}^s$(*i.e.*, each row contains one $s-$dimensional point).

`ctrs` $N \times s$ matrix representing a set of N centers in $\mathbb{R}^s$ (one center per row).

Output:

`DM` $M \times N$ matrix whose i,j position contains the Euclidean distance between the $i-$th data site and $j-$th center.

Purpose: Solve ill-conditioned symmetric positive definite linear systems. It is active if the data sites are more than 1000.

Call:

```
x = Riley(A,b)
```

Input parameters:

`A`	Symmetric positive definite matrix.
`b`	Vector with constant terms.

Output:

`x`	Solution of the linear system.

Purpose: Find a local minimum of a function in a bounded interval.

Call:

```
[x, fx, exitflag, output] = ...
  rbf_minsearch(fun, xmin, xmax, options, fparam_DM_data, ...
    fparam_rbf, fparam_dvalues)
```

Input parameters:

`fun`	Function for which we want a local minimum.	
`xmin`,`xmax`	Bounds of the interval in which we search a minimum.	
`options`	Struct currently not optional with the elements	
	`TolX`	Tolerance on the distance from the minimum fixed to 10^{-8}.
	`MaxIter`	Maximum number of iterations fixed to infinity.
	`MaxFunEvals`	Maximum number of evaluations of the function fixed to infinity.
	`PrintToScreen`	If true, print information to screen. It is currently set to false.
`fparam_DM_data`	Distance matrix.	
`fparam_rbf`	Radial basis function.	
`fparam_dvalues`	Data values.	

Output:

`x`	Local minimum of the function.	
`fx`	Value of the function in `x`.	
`exitflag`	It gives information about the convergence and can take three values:	
	1	Convergence.
	0	Reached maximum number of iteration, function evaluations.
	-1	Not convergence/error.
`output`	It is possible to have an output with all the information about the convergence of the method.	

5.6 Data Files

Currently the toolbox contains two test data sets, illustrated in Section 5.7. The command

```
load('data1','dsites','dvalues')
```

makes the arrays `dsites` 150×2 and `dvalues` 150×1 available in the workspace,*i.e.*, $N = 150$ data sites in $s = 2$ dimensions and the related measurements. The design sites stored in `dsites` are sampled in the two-dimensional area $[0, 100]^2$.
The command

```
load('dataFranke','dsites','dvalues')
```

makes the arrays `dsites` 100×2 and `dvalues` 100×1 available in the workspace, *i.e.*, $N = 100$ data sites in $s = 2$ dimensions and the related values of the Franke bivariate test function

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

The design sites stored in `dsites` are sampled in the two-dimensional area $[0, 1]^2$.

5.7 Example of usage

An example of simple usage of the two most important functions in the toolbox, namely `rmfmodel` and `rbfapprox`, is presented here. The example shows how one can get a surface approximation to a given data set. We start by loading the data set `data1` provided with the toolbox, cf. Section 5.6,

```
load('data1','dsites','dvalues')
```

Now the 150×2 array `dsites` and 150×1 array `dvalues` are present in the workspace. We choose the `gaussian` RBF, centers coinciding with data sites (`ctrs = dsites`) and the mode as the value picked in case of repeated points. We want to find a "good" shape parameter in the interval $[0, 10]$. We are now ready to make the model by calling `rbfmodel`,

```
cmodel = rbfmodel(dsites, dvalues, gaussian, [], ctrs, 0, 10, fmode);
```

From the returned result we can extract information about the generated model. For instance, the shape parameter that will be used is

```
cmodel.eps = 4.016625
```

Having the model been stored in the structure array `cmodel` we may use it for prediction at new (untried) sites. We generate a grid of points on which to evaluate the predictor `rbfapprox`. We choose a 40×40 mesh of points distributed equidistantly in the area $[0, 100]^2$ covered by the design sites, and call the predictor with the mesh points and the `cmodel`,

```
neval = 40; grid = linspace(0,100,neval);
[xe,ye] = meshgrid(grid); epoints = [xe(:) ye(:)];
Pf = rbfapprox(epoints, cmodel);
```

Since we have less than 1000 data sites, `PfRiley` would be identical to `Pf`, thus we do not ask for this parameter as output. The returned vector `Pf` contains the predicted values at the mesh. In order to plot the result we reshape `Pf` to match the grid and use `xe` and `ye`, which contain the reshaped coordinates of the evaluation points:

```
f = scf(1)
mesh(xe,ye,matrix(Pf,neval,neval));
f.color_map = oceancolormap(5);
a = gca()
a.font_size = 4;
```

The resulting plot is shown in Figure 5.1.

To give another example we can load the data set `dataFranke` present in the toolbox (see Section 5.6) and therefore approximate the Franke bivariate function:

```
load('dataFranke','dsites','dvalues')
```

Now the 100×2 array `dsites` and 100×1 array `dvalues` are present in the workspace. We choose the `QuadraticMatern` RBF, centers coinciding with data sites (`ctrs = dsites`) and the mode as the value picked in case of repeated points. In this case we want to find a "good" shape parameter in the interval $[2, 3]$. We are now ready to make the model by calling `rbfmodel`,

```
cmodel = rbfmodel(dsites, dvalues, QuadraticMatern, [], ctrs, 2, 3, fmode);
```

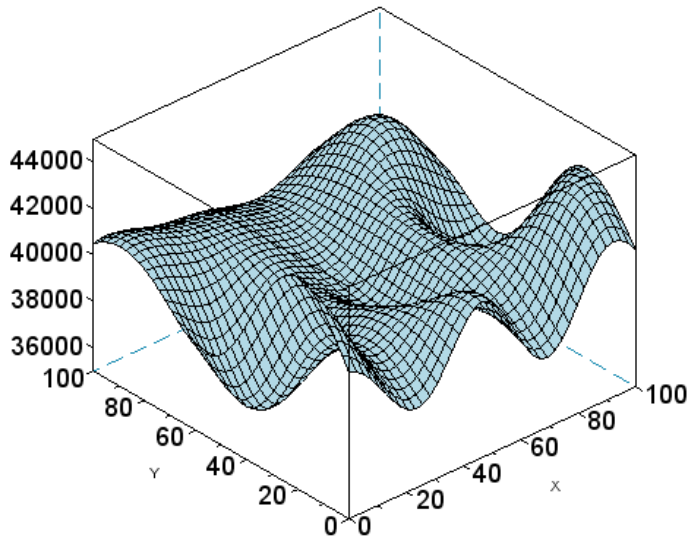


Figure 5.1: Mesh plot of the predicted values at the grid points

The shape parameter that will be used is `cmodel.eps = 2.834928`.
We may now use `cmodel` for prediction at new (untried) sites. As in the first example we generate a 40×40 grid of points on which to evaluate the predictor `rbfapprox`, with the differende that now the area is $[0, 1]^2$. Then we can call the predictor with the mesh points and `cmodel`,

```
neval = 40; grid = linspace(0,1,neval);
[xe,ye] = meshgrid(grid); epoints = [xe(:) ye(:)];
Pf = rbfapprox(epoints, cmodel);
```

Since, in this case, we know the real values of the Franke function in the evaluation points, we can also compute the maximum error and the root mean square error:

```
exact = Franke(epoints);
maxerr = norm(Pf-exact,'inf');
rms_err = norm(Pf-exact)/neval;
printf('Maximum error: %e\n', maxerr)
printf('RMS error:     %e\n', rms_err)
```

getting

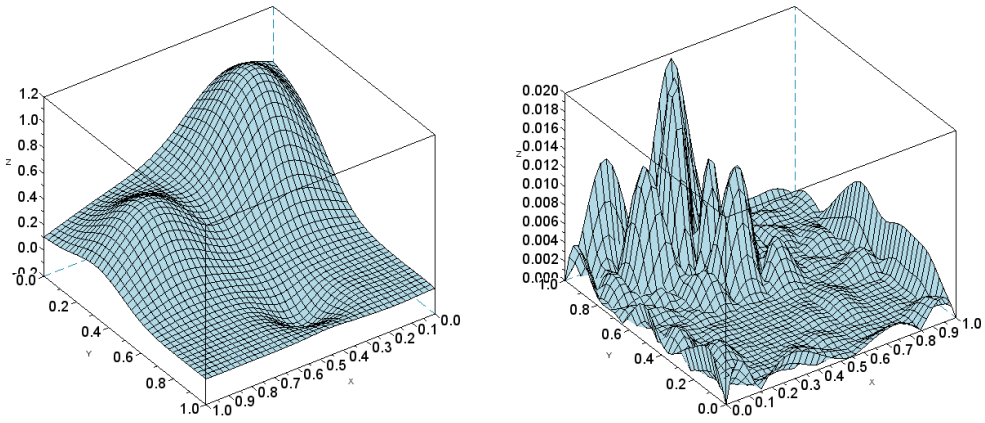


Figure 5.2: Mesh plot of the predicted values of the Franke function and the absolute approximation error.

```
Maximum error: 1.979431e-002
RMS error:     3.176449e-003
```

We are now ready to reshape `Pf` and the absolute approximation error `abs(Pf-exaxt)` and to plot them as in Figure 5.2:

```
f = scf(1)
subplot(1,2,1)
mesh(xe,ye,matrix(Pf,neval,neval));
f.color_map = oceancolormap(5);
a = gca()
a.font_size = 4;
subplot(1,2,2)
mesh(xe,ye,matrix(abs(Pf-exact),neval,neval));
f.color_map = oceancolormap(5);
a = gca()
a.font_size = 4;
```

5.8 How to install the package

In order to install the package the following steps should be done:

1. Download the package "RBFtoolbox.zip" from the Openeering site www.openeering.com;

2. Unzip the file in a working directory (e.g. "D:\scilabpackages");
3. Open Scilab and move to the package directory (in our case "D:\cilabpackages\RBFtoolbox") using the command `cd ''D:\scilabpackages\RBFtoolbox''`;
4. Build the package using the command `exec builder.sce`;
5. Load the package using the command `exec loader.sce`;
6. Try demo clicking on the Scilab demonstration icon in the toolbar and select demo from the RBF toolbox

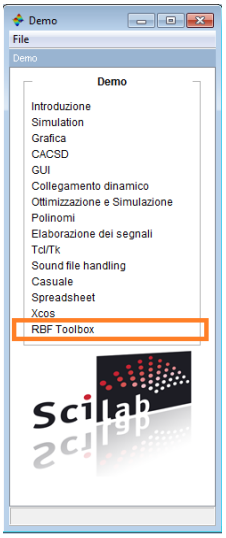


Figure 5.3: Scilab demonstration.

7. Try to run the demo Franke, for example
8. Check in the help, the RBFtoolbox is available for further details.

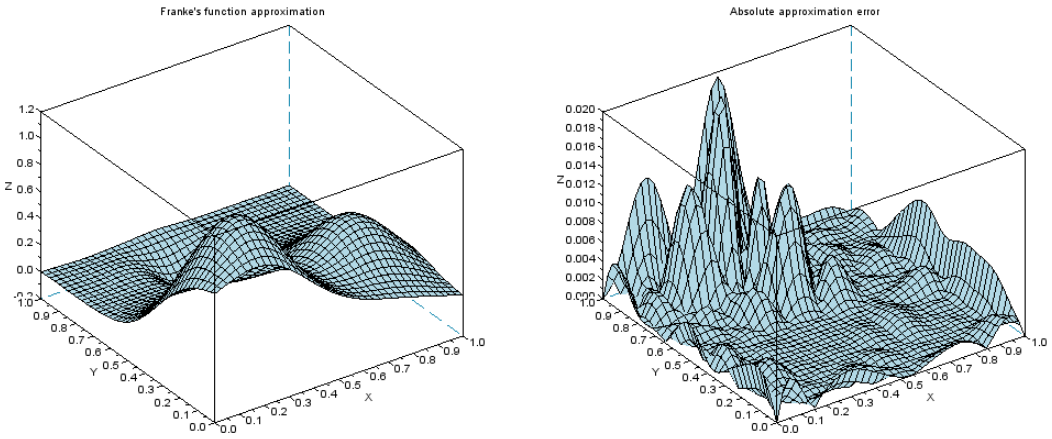


Figure 5.4: RBFtoolbox Franke demo.

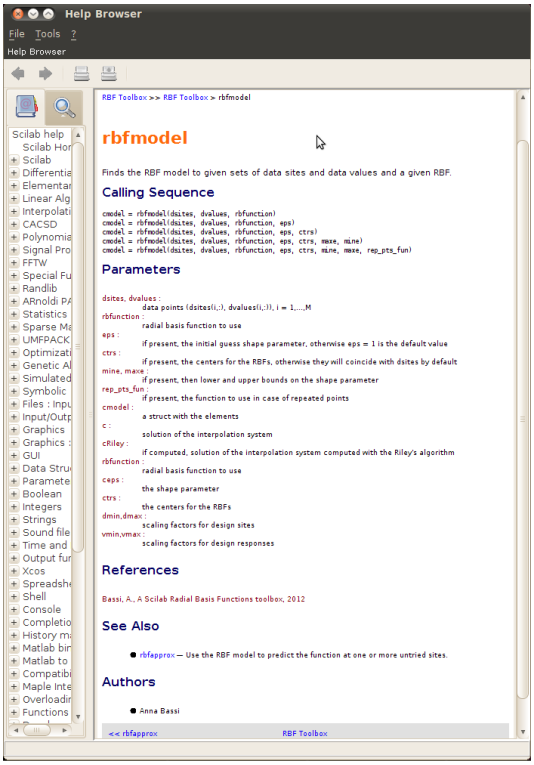


Figure 5.5: Help for `rbfmodel` function.

Bibliography

[1] Brent, R. P., Algorithms for Minimization Without Derivatives, Prenctice-Hall, 1973.

[2] Fasshauer, G., Meshfree Approximation Methods with Matlab, World Scientific Publishers, 2007.

[3] Golub, G. H., Numerical methods for solving linear least-squares problems, 1965.

[4] Kincaid, D. R., Cheney, E. W., Numerical Analysis, American Mathematical Society, 2002.

[5] Riley, J. D., Solving systems of linear equations with a positive definite, symmetric, but possibly ill-conditioned matrix, Mathematical Tables and Other Aids to Computation 9/51, pp. 96101, 1955.

[6] Rippa, S., An algorithm for selecting a good value for the parameter c in radial basis function interpolation, Adv. in Comput. Math. 11, pp. 193210, 1999.

[7] Sharma, N., Gobbert, M. K., A comparative evaluation of Matlab, GNU Octave, Freemat, and Scilab for research and teaching, Technical Report HPCF-2010-7, www.umbc.edu/hpcf, 2010.

[8] Wendland, H., Scattered Data Approximation, Cambridge University Press, 2005.