Chapter 7

SURROGATE MODELS

7.1 Introduction

A Taylor series expansion of a function allows us to approximate a function $f(\mathbf{x})$ at any point $\mathbf{x}$, based solely on information about the function at a single point $\mathbf{x}^i$. Here, information about the function implies zero order, first order, second order and higher order information of the function at $\mathbf{x}^i$. The higher the order of information included in a Taylor series representation of a function, the higher the accuracy of the approximation distant from $\mathbf{x}^i$. However, higher order information for multivariate functions grows exponentially in dimensionality, i.e. the gradient vector constitutes n values, the Hessian matrix is expressed by n^2 values and the 3rd derivative is comprised of n^3 values. In practice, zero order and first order information about a problem is usually computable and convenient to store, while second order information is usually not readily available for engineering problems and needs to be inferred from first order information as described in Section 2.4.

Surrogate modelling offers an alternative approach to Taylor series for constructing approximations of functions. Instead of constructing approximations based on ever higher and higher order information at a single point, surrogate modelling approximates functions using lower order information at numerous points in the domain of interest. The advantage of such an approach is that it is (i) computationally inexpensive to

J.A. Snyman and D.N. Wilke, *Practical Mathematical Optimization,*
Springer Optimization and Its Applications 133,
https://doi.org/10.1007/978-3-319-77586-9_7

approximate zero and first order information of the function at additional points in the domain, and that (ii) lower order information can be computed in parallel on distributed computing platforms. Hence, the approximation functions can be exhaustively optimized, while the computationally demanding evaluations of the actual function can be distributed over multiple cores and computers. It is not surprising that surrogate modelling is the preferred strategy to solve computationally demanding multidisciplinary engineering design problems as highlighted by Forrester et al. (2008). However, as information grows exponentially with the order of information for a multivariate problem, so too does the design space grow exponentially with problem dimensionality. This is referred to as, the *curse of dimensionality*, as phrased by Bellman (1957). This limits surrogate modelling to lower-dimensional problems in the same way that Taylor series approximations are limited to lower order information for higher-dimensional problems.

Formally, a surrogate model approximates a non-linear function $f(\mathbf{x})$, when information about $f(\mathbf{x})$ is known at m discrete locations $\mathbf{x}^i$, $i = 1, \ldots, m$. The information is usually limited to zero order information, i.e. only function values as described by Hardy (1971, 1990); Franke (1982); Dyn et al. (1986); Khuri and Mukhopadhyay (2010). More recently both zero and first order information have been considered more readily in the construction of surrogate models, Hardy (1975, 1990); Morris et al. (1993); Chung and Alonso (2001); Lauridsen et al. (2002). Lastly, Wilke (2016) proposed the construction of surrogate models using only first order information. This allows for smooth surrogate approximations of piecewise smooth discontinuous functions as will be shown at the end of this chapter.

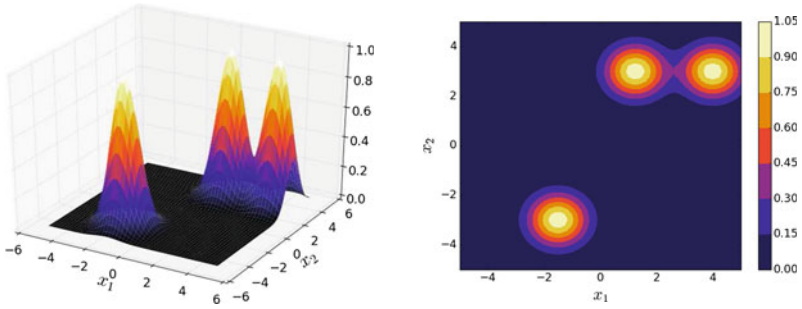
7.2 Radial basis surrogate models

A number of surrogate models are available to approximate a non-linear function $f(\mathbf{x})$. Hardy (1971) pioneered radial basis functions by approximating a non-linear function, $f(\mathbf{x})$, as a linear combination of p chosen non-linear basis functions $\phi_j(\mathbf{x}, \mathbf{x}_c^j)$, $j = 1, \ldots, p$, that are centered

around p spatial points $\mathbf{x}_c^j$, which is conveniently expressed by

$$f(\mathbf{x}) \approx \sum_{j=1}^{p} w_j \phi_j(\mathbf{x}, \mathbf{x}_c^j) = \tilde{f}(\mathbf{x}). \tag{7.1}$$

The non-linear basis functions, $\phi_j(\mathbf{x}, \mathbf{x}_c^j),\ j = 1, \ldots, p$, are usually chosen to be of identical form but centered around distinct spatial points $\mathbf{x}_c^j$. For examples of radial basis functions refer to Table 7.1. Typically Figures 7.1 (a) and (b) depict three basis functions centered about three points in a two-dimensional design domain.



(a) (b)

Figure 7.1: Illustration of three radial basis functions centered around three spatial locations, namely, $x_1 = -1.5$ and $x_2 = -3$, $x_1 = 1.25$ and $x_2 = 3$, and $x_1 = 4$ and $x_2 = 3$

A significant benefit of this approach is that all the non-linearity of $f(\mathbf{x})$ is approximated by the non-linear basis functions $\phi_j(\mathbf{x})$, while the approximation is linear in the weights w_j that need to be estimated. Hence, w_j can be obtained by merely solving a linear problem.

7.2.1 Zero order only radial basis surrogate models

Solving for the weights $w_j,\ j = 1, \ldots, p$ from only zero order (zo) information requires the function, $f(\mathbf{x}^j)$, to be evaluated at least for $m \geq p$ distinct designs $\mathbf{x}^j,\ j = 1, \ldots, m$. The *response* surface $\tilde{f}(\mathbf{x})$ is then required to recover the actual function value at the $\mathbf{x}^j,\ j = 1, \ldots, m$ designs, i.e. $\tilde{f}(\mathbf{x}^j) \approx f(\mathbf{x}^j)$. By choosing the number of designs and

basis functions to be equal, i.e. $m = p$, we recover an *interpolation* surface, i.e. $\tilde{f}(\mathbf{x}^j) = f(\mathbf{x}^j),\ j = 1, \ldots, m,$. When we have fewer basis functions than design vectors, i.e. $p < m$, we recover a *regression* surface $\tilde{f}(\mathbf{x}^j) \approx f(\mathbf{x}^j),\ j = 1, \ldots, m$, that requires a least squares problem to be solved to recover the weights. In general interpolation surfaces are preferred when only a few design vectors are available, while regression surfaces are favoured when the design domain is densely sampled.

In general, by choosing p basis functions and m design vectors, results in p unknowns to be solved from m equations

$$\begin{aligned} f(\mathbf{x}^1) &= \sum_{j=1}^{p} w_j \phi_j(\mathbf{x}^1, \mathbf{x}_c^j) = \tilde{f}(\mathbf{x}^1) \\ f(\mathbf{x}^2) &= \sum_{j=1}^{p} w_j \phi_j(\mathbf{x}^2, \mathbf{x}_c^j) = \tilde{f}(\mathbf{x}^2) \\ &\vdots \\ f(\mathbf{x}^m) &= \sum_{j=1}^{p} w_j \phi_j(\mathbf{x}^m, \mathbf{x}_c^j) = \tilde{f}(\mathbf{x}^m). \end{aligned} \tag{7.2}$$

This can be rewritten in block matrix form

$$\mathbf{R}_{zo}\mathbf{w}^{zo} = \mathbf{r}^{zo}, \tag{7.3}$$

with

$$\mathbf{R}_{zo} = \begin{bmatrix} \phi_1(\mathbf{x}^1, \mathbf{x}_c^1) & \phi_2(\mathbf{x}^1, \mathbf{x}_c^2) & \ldots & \phi_m(\mathbf{x}^1, \mathbf{x}_c^p) \\ \vdots & & & \\ \phi_1(\mathbf{x}^m, \mathbf{x}_c^1) & \phi_2(\mathbf{x}^m, \mathbf{x}_c^2) & \ldots & \phi_m(\mathbf{x}^m, \mathbf{x}_c^p) \end{bmatrix},$$

$$\mathbf{w}^{zo} = \begin{bmatrix} w_1^{zo} \\ w_2^{zo} \\ \vdots \\ w_p^{zo} \end{bmatrix}, \mathbf{r}^{zo} = \begin{bmatrix} f(\mathbf{x}^1) \\ \vdots \\ f(\mathbf{x}^m) \end{bmatrix},$$

to obtain an m by p linear system of equations. For $p = m$ the system can be solved directly. However, $p < m$ yields an overdetermined system of equations to be solved in a least squares sense. This is achieved by

pre-multiplying (7.3) by $\mathbf{R}_{zo}^{\mathrm{T}}$ to obtain the following $p \times p$ linear system of equations

$$\mathbf{R}_{zo}^{\mathrm{T}}\mathbf{R}_{zo}\mathbf{w}^{zo} = \mathbf{R}_{zo}^{\mathrm{T}}\mathbf{r}^{zo}, \tag{7.4}$$

from which $\mathbf{w}^{zo}$ can be solved for.

For convenience, when choosing $p = m$, the m designs $\mathbf{x}^j,\ j = 1, \ldots, m$ are usually chosen to coincide with the p basis function centers, i.e. $\mathbf{x}_c^j = \mathbf{x}^j,\ j = 1, \ldots, m$.

7.2.2 Combined zero and first order radial basis surrogate models

Solving for the weights $w_j,\ j = 1, \ldots, p$ from both zero and first order information, i.e. mixed order (mo), requires the function, $f(\mathbf{x}^j)$, and gradient of the function, $\nabla f(\mathbf{x}^j)$, to be evaluated at m distinct designs $\mathbf{x}^j,\ j = 1, \ldots, m$. The response surface $\tilde{f}(\mathbf{x})$ is then required to recover the actual function value and gradient at the $\mathbf{x}^j,\ j = 1, \ldots, m$ designs, i.e. $\tilde{f}(\mathbf{x}^j) \approx f(\mathbf{x}^j)$ and $\nabla\tilde{f}(\mathbf{x}^j) \approx \nabla f(\mathbf{x}^j)$. By choosing m designs for $p = m$ basis functions we recover a regression surface, i.e. $\tilde{f}(\mathbf{x}^j) \approx f(\mathbf{x}^j),\ j = 1, \ldots, m$, and $\nabla\tilde{f}(\mathbf{x}^j) \approx \nabla f(\mathbf{x}^j),\ j = 1, \ldots, m$. A least squares problem is then to be solved to recover the weights.

In general, by choosing p basis functions and m design vectors, each of dimension n, we now have p unknowns to be solved for from $m(n + 1)$

equations:

$$\begin{aligned}
f(\mathbf{x}^1) &= \sum_{j=1}^{p} w_j \phi_j(\mathbf{x}^1, \mathbf{x}_c^j) = \tilde{f}(\mathbf{x}^1), \\
\nabla f(\mathbf{x}^1) &= \sum_{j=1}^{p} w_j \frac{\partial \phi_j(\mathbf{x}^1, \mathbf{x}_c^j)}{\partial \mathbf{x}} = \nabla \tilde{f}(\mathbf{x}^1), \\
f(\mathbf{x}^2) &= \sum_{j=1}^{p} w_j \phi_j(\mathbf{x}^2, \mathbf{x}_c^j) = \tilde{f}(\mathbf{x}^2), \\
\nabla f(\mathbf{x}^2) &= \sum_{j=1}^{p} w_j \frac{\partial \phi_j(\mathbf{x}^2, \mathbf{x}_c^j)}{\partial \mathbf{x}} = \nabla \tilde{f}(\mathbf{x}^2), \\
&\vdots \\
f(\mathbf{x}^m) &= \sum_{j=1}^{p} w_j \phi_j(\mathbf{x}^m, \mathbf{x}_c^j) = \tilde{f}(\mathbf{x}^m), \\
\nabla f(\mathbf{x}^m) &= \sum_{j=1}^{p} w_j \frac{\partial \phi_j(\mathbf{x}^m, \mathbf{x}_c^j)}{\partial \mathbf{x}} = \nabla \tilde{f}(\mathbf{x}^m),
\end{aligned} \tag{7.5}$$

which can be rewritten in block matrix form to obtain

$$\mathbf{R}_{mo}\mathbf{w}^{mo} = \mathbf{r}^{mo}, \tag{7.6}$$

with

$$\mathbf{R}_{mo} = \begin{bmatrix}
\phi_1(\mathbf{x}^1, \mathbf{x}_c^1) & \phi_2(\mathbf{x}^1, \mathbf{x}_c^2) & \dots & \phi_m(\mathbf{x}^1, \mathbf{x}_c^p) \\
\frac{\partial \phi_1(\mathbf{x}^1, \mathbf{x}_c^1)}{\partial \mathbf{x}} & \frac{\partial \phi_2(\mathbf{x}^1, \mathbf{x}_c^2)}{\partial \mathbf{x}} & \dots & \frac{\partial \phi_m(\mathbf{x}^1, \mathbf{x}_c^p)}{\partial \mathbf{x}} \\
\vdots & & & \\
\phi_1(\mathbf{x}^m, \mathbf{x}_c^1) & \phi_2(\mathbf{x}^m, \mathbf{x}_c^2) & \dots & \phi_m(\mathbf{x}^m, \mathbf{x}_c^p) \\
\frac{\partial \phi_1(\mathbf{x}^m, \mathbf{x}_c^1)}{\partial \mathbf{x}} & \frac{\partial \phi_2(\mathbf{x}^m, \mathbf{x}_c^2)}{\partial \mathbf{x}} & \dots & \frac{\partial \phi_m(\mathbf{x}^m, \mathbf{x}_c^p)}{\partial \mathbf{x}}
\end{bmatrix},$$

$$\mathbf{w}^{mo} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}, \mathbf{r}^{mo} = \begin{bmatrix} f(\mathbf{x}^1) \\ \nabla f(\mathbf{x}^1) \\ \vdots \\ f(\mathbf{x}^m) \\ \nabla f(\mathbf{x}^m) \end{bmatrix},$$

to obtain a $m(n+1)$ by p overdetermined linear system of equations. The overdetermined system of equations can be solved in a least squares sense by pre-multiplying (7.6) by $\mathbf{R}_{mo}^{\mathrm{T}}$ to obtain the following $p \times p$ linear system of equations

$$\mathbf{R}_{mo}^{\mathrm{T}} \mathbf{R}_{mo} \mathbf{w}^{mo} = \mathbf{R}_{mo}^{\mathrm{T}} \mathbf{r}^{mo}, \tag{7.7}$$

from which $\mathbf{w}^{mo}$ can be solved for.

7.2.3 First order only radial basis surrogate models

Solving for the weights $w_j,\ j = 1, \ldots, p$ from only first order (fo) or gradient only information requires that the gradient $\nabla f(\mathbf{x})$ of the non-linear function to be evaluated at m distinct designs $\mathbf{x}^j,\ j = 1, \ldots, m$. The response surface $\tilde{f}(\mathbf{x})$ is then required to recover the actual gradient at the $\mathbf{x}^j,\ j = 1, \ldots, m$ designs, i.e. $\nabla \tilde{f}(\mathbf{x}^j) \approx \nabla f(\mathbf{x}^j)$. By choosing m designs and $p = m$ basis functions we recover an interpolation surface for the gradient, i.e. $\nabla \tilde{f}(\mathbf{x}^j) = \nabla f(\mathbf{x}^j),\ j = 1, \ldots, m$, only for univariate functions.

In general, higher-dimensional functions result in regression response surfaces, i.e. $\nabla \tilde{f}(\mathbf{x}^j) \approx \nabla f(\mathbf{x}^j)$, since we only have $p = m$ weights to recover $m \times n = mn$ gradient components, where n is the dimension of the design vector. The resulting mn equations

$$\begin{aligned}
\nabla f(\mathbf{x}^1) &= \sum_{j=1}^{p} w_j \frac{\partial \phi_j(\mathbf{x}^1, \mathbf{x}_c^j)}{\partial \mathbf{x}} = \nabla \tilde{f}(\mathbf{x}^1) \\
\nabla f(\mathbf{x}^2) &= \sum_{j=1}^{p} w_j \frac{\partial \phi_j(\mathbf{x}^2, \mathbf{x}_c^j)}{\partial \mathbf{x}} = \nabla \tilde{f}(\mathbf{x}^2) \\
&\vdots \\
\nabla f(\mathbf{x}^m) &= \sum_{j=1}^{p} w_j \frac{\partial \phi_j(\mathbf{x}^m, \mathbf{x}_c^j)}{\partial \mathbf{x}} = \nabla \tilde{f}(\mathbf{x}^m),
\end{aligned} \tag{7.8}$$

can be rewritten in block matrix form to obtain

$$\mathbf{R}_{fo} \mathbf{w}^{fo} = \mathbf{r}^{fo}, \tag{7.9}$$

with

$$\mathbf{R}_{fo} = \begin{bmatrix} \frac{\partial \phi_1(\mathbf{x}^1,\mathbf{x}_c^1)}{\partial \mathbf{x}} & \frac{\partial \phi_2(\mathbf{x}^1,\mathbf{x}_c^2)}{\partial \mathbf{x}} & \cdots & \frac{\partial \phi_m(\mathbf{x}^1,\mathbf{x}_c^p)}{\partial \mathbf{x}} \\ \vdots & & & \\ \frac{\partial \phi_1(\mathbf{x}^m,\mathbf{x}_c^1)}{\partial \mathbf{x}} & \frac{\partial \phi_2(\mathbf{x}^m,\mathbf{x}_c^2)}{\partial \mathbf{x}} & \cdots & \frac{\partial \phi_m(\mathbf{x}^m,\mathbf{x}_c^p)}{\partial \mathbf{x}} \end{bmatrix},$$

$$\mathbf{w}^{fo} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}, \mathbf{r}^{fo} = \begin{bmatrix} \nabla f(\mathbf{x}^1) \\ \vdots \\ \nabla f(\mathbf{x}^m) \end{bmatrix},$$

giving an mn by m overdetermined linear system of equations. This overdetermined system of equations can be solved in a least squares sense by pre-multiplying (7.9) by $\mathbf{R}_{fo}^{\mathrm{T}}$ to obtain the following $m \times m$ linear system of equations

$$\mathbf{R}_{fo}^{\mathrm{T}}\mathbf{R}_{fo}\mathbf{w}^{fo} = \mathbf{R}_{fo}^{\mathrm{T}}\mathbf{r}^{fo}, \tag{7.10}$$

from which $\mathbf{w}^{fo}$ is to be solved for. However, $\mathbf{R}_{fo}^{\mathrm{T}}\mathbf{R}_{fo}$ is singular as an approximated surrogate from first order only information has infinite representations since any constant added to the surrogate leaves the gradient of the surrogate unchanged. This can be addressed by adding at least one equation that enforces the function value at a design to (7.10). Alternatively the minimum norm solution for the least squares system (7.10) can be computed. For convenience, the m designs $\mathbf{x}^j,\ j = 1,\ldots,m$ are usually chosen to coincide with the chosen $p = m$ basis function centers, i.e. $\mathbf{x}_c^j = \mathbf{x}^j,\ j = 1,\ldots,m$.

7.3 Basis functions

Numerous radial basis functions $\phi_j(r_j)(\mathbf{x})$ have been proposed with the most popular global support basis functions listed in Table 7.1. The shape parameter ϵ is in general unknown and needs to be determined as will be discussed in the next section. Smooth basis functions are preferred when constructing surrogate models for optimization applications, as they are everywhere differentiable. A preferred choice is the Gaussian basis function,

$$\phi_j(\mathbf{x},\mathbf{x}_c^j) = \phi_j(r_j(\mathbf{x})) = e^{(-\epsilon r_j(\mathbf{x})^2)}. \tag{7.11}$$

Name	Abbreviation	Equation
Gaussian	GA	$\phi_j(r_j(\mathbf{x})) = e^{-\epsilon r_j(\mathbf{x})^2}$
Exponential	EXP	$\phi_j(r_j(\mathbf{x})) = e^{-\epsilon r_j(\mathbf{x})}$
Multiquadric	MQ	$\phi_j(r_j(\mathbf{x})) = \sqrt{1+(\epsilon r_j(\mathbf{x}))^2}$
Inverse quadratic	IQ	$\phi_j(r_j(\mathbf{x})) = \frac{1}{1+(\epsilon r_j(\mathbf{x}))^2}$
Inverse multiquadric	IMQ	$\phi_j(r_j(\mathbf{x})) = \frac{1}{\sqrt{1+(\epsilon r_j(\mathbf{x}))^2}}$

Table 7.1: Radial basis functions $\phi_j(r_j(\mathbf{x}))$ with $r_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}^j\|$

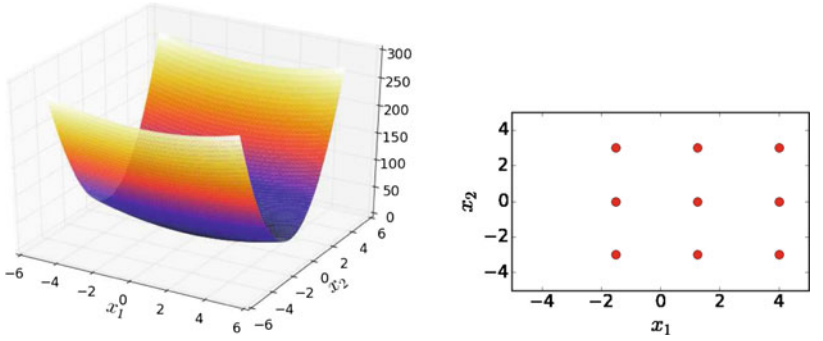
The gradient is given by

$$\nabla\phi_j(r_j(\mathbf{x})) = \frac{\partial\phi_j}{\partial r_j}\frac{\partial r_j}{\partial \mathbf{x}} = \left(-2\epsilon r_j e^{(-\epsilon r_j^2)}\right)\left(\frac{1}{2r_j}2(\mathbf{x}-\mathbf{x}_c^j)\right), \qquad (7.12)$$

and tends to $\mathbf{0}$ as $r_j(\mathbf{x}) \to 0$. However, as (7.12) is prone to numerical instabilities as $r_j(\mathbf{x}) \to 0$, it is required to apply L'Hospital's rule to ensure that numerically $\nabla\phi_j(r_j(\mathbf{x}))$ indeed evaluates to $\mathbf{0}$ at $r_j(\mathbf{x}_c^j)$.

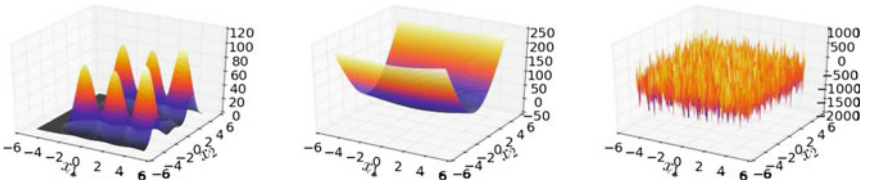
7.3.1 Shape parameter

The shape parameter ϵ needs to be estimated. It determines the radius of the domain over which the basis function has significant influence. For example, consider the Gaussian basis function (7.11), for ϵ chosen large and unit r_j, $-\epsilon r_j^2$ evaluates to a large negative value of which the exponential is close to zero. Consequently, the larger ϵ the smaller the domain over which the basis function has a significant influence. The choice of ϵ is therefore of utmost importance, with smaller ϵ preferred, but choosing ϵ too small will result in severe numerical ill-conditioning for finite precision computing.



(a) (b)

Figure 7.2: (a) Two-dimensional quadratic function evaluated at (b) nine points in the two-dimensional design domain



(a) (b) (c)

Figure 7.3: Radial basis approximation surfaces constructed using (a) $\epsilon = 10^0$, (b) $\epsilon = 10^{-1}$ and (c) $\epsilon = 10^{-5}$

Consider the construction of zero-order Gaussian radial basis surrogate approximations of the quadratic function, $f(\mathbf{x}) = x_1^2 + 10x_2^2$, depicted in Figure 7.2 (a) that is evaluated at nine points as shown in Figure 7.2 (b).

The construction done for three values of ϵ are depicted in Figures 7.3 (a)–(c). The results clearly indicate the influence of the shape parameter on the constructed radial basis function approximation surfaces. Choosing ϵ too large results in locally compact support that fails to capture smooth trends over large domains as shown in Figure 7.3 (a). Ill-conditioning is evident in Figure 7.3 (c) in which the numerical solution for the weights have broken down resulting in severe noise. Lastly, choosing an appro-

priate shape parameter gives the ability to capture the actual function using spatially distributed information as illustrated in Figure 7.3 (b).

In general, ϵ is computed using k-fold cross validation (Kohavi (1995)). In k-fold cross validation the m design vectors are randomly partitioned into k equal sized subsets, i.e. each subset containing $\frac{m}{k}$ design vectors. Of the k subsets, the first is retained to test the model (test subset), while the union of the remaining $k-1$ subsets (the training subsets) is used to construct the response surface. Once, the response surface has been constructed, using only the training subset, the model is used to predict the response at the design vectors in the test subset. Since the responses at the designs in the test subset are known an error can be computed that captures the difference between the predicted and actual response. Here, the sum of the difference squared is usually computed. This process is repeated by choosing the next subset as the test subset, while the union of the remaining $k-1$ subsets again define the training subset until all subsets have been used as a test subset, i.e. this process is repeated k times. All the errors over the k test subsets are averaged to define the k-fold cross validation error (k-CVE). The k-CVE is then computed for different choices of ϵ to find the ϵ^* that minimizes k-CVE error. Afterwards, ϵ^* is used to construct the surrogate, usually using all m points. A typical choice for k is between 5 and 20, while choosing $k = m$ results in leave-one-out cross validation (LOOCV). This forms the basis for the predicted residual error sum of squares (PRESS) statistic proposed by Allen (1974).

The process of constructing surrogate models using the k-CVE is listed in Algorithm 7.1.

7.4 Numerical examples

To demonstrate the implications and utility of using only zero or only first order information in constructing surrogate models, consider the two test functions depicted in Figure 7.4 (a) and (b). They are the smooth continuous quadratic function

$$f(\mathbf{x}) = \sum_{i=1}^{n} 10^{i-1} x_{i-1}^2, \tag{7.13}$$

Algorithm 7.1 Radial basis surrogate model.

Initialization: Select a radial basis function and associated initial shape parameter ϵ_0. Choose an integer value for k and select the number of design vectors m (a multiple of k) in the design of experiments. Randomly partition the m design vectors into k subsets of equal size. These k subsets are then used to compute the k-fold cross validation error (k-CVE). Set $l = 0$ and perform the following steps:

1. **Design of Experiments:** Identify m design vectors $\mathbf{x}^i,\ i = 1, \ldots, m$.

2. **Evaluate Designs:** For each design, compute the function value $f_i = f(\mathbf{x}^i),\ i = 1, \ldots m$ and/or gradient vector $\nabla f^i = \nabla f(\mathbf{x}^i),\ i = 1, \ldots m$.

3. **Trial Surrogate model:** For $\epsilon = \epsilon_l$ construct k trial surrogate models by solving for $\mathbf{w}$ from (7.4), (7.7) or (7.10) using successively, each of the k subsets containing $\frac{m}{k}$ design vectors as the test subset, and the union of the remaining $k - 1$ subsets as training set. For each of the k surrogate models use its corresponding test set and compute the test set error. Then compute the k-CVE as the average of the k test set errors.

4. **Update ϵ_l:**

   (a) Set $l := l + 1$

   (b) Update ϵ_l using a minimization strategy to reduce the k-CVE.

   (c) If ϵ_l has converged within an acceptable tolerance then set $\epsilon^* = \epsilon_l$ and go to Step 5, else go to Step 3.

5. **Construct Surrogate Model:** Construct the surrogate model by solving for $\mathbf{w}$ from (7.4), (7.7) or (7.10) using $\epsilon = \epsilon^*$. Instead of only using the training set of design vectors it is often advised to use all m design vectors to construct the final surrogate model. A typical choice for k is between 5 and 20.

depicted in Figure 7.4 (a) with $n = 2$, and the piece-wise smooth step discontinuous quadratic function

$$f(\mathbf{x}) = \sum_{i=2}^{n} 10^{i-2} x_{i-2}^2 + 10^{i-1} x_{i-1}^2 + a(\text{sign}(x_{i-2})) - b(\text{sign}(x_{i-1})) + c(\text{sign}(x_{i-2}))(\text{sign}(x_{i-1})) \quad (7.14)$$

depicted in Figure 7.4 (b) with $n = 2$, $a = 100$, $b = 50$ and $c = 133$. It is important to note that the piece-wise smooth step discontinuous quadratic function is the same quadratic function given by (7.13) with step discontinuities imposed that are controlled by the coefficients a, b and c. Hence, the first *associated partial derivatives* w.r.t. x_1 and x_2 for both the smooth quadratic function and step discontinuous quadratic function are the same and respectively depicted in Figures 7.4 (c) and (d). The functions are evaluated at 3×3 designs as depicted in Figure 7.2 (b) to obtain the zero or first order information required to construct the surrogate models. Gaussian radial basis functions are used in the models and $k = m$ is specified for computing the CVE. Only zero order information or only first order information are considered.

The constructed zero order only and first order only (with the exception of enforcing $f = 0$ at $x_1 = 0$, $x_2 = 0$) surrogate models for the smooth quadratic function are depicted in Figures 7.5 (a) and (b). In addition, the first order partial derivatives w.r.t. x_1 and x_2 of the surrogate models are respectively depicted in Figures 7.6 (a) and (b) and Figures 7.6 (c) and (d) over the domain -5 to 5 for both variables.

It is evident that the two constructed surfaces, depicted in Figures 7.5 (a) and (b), are nearly identical. Similarly, the estimated first order partial derivatives for the zero order only and first order only constructed surrogate models w.r.t. x_1 and x_2 are respectively depicted in Figures 7.6 (a) and (b) and Figures 7.6 (c) and (d). The approximated partial derivatives are nearly identical to the actual partial derivatives depicted in Figures 7.4 (c) and (d).

For the piece-wise smooth step discontinuous quadratic function, the constructed zero order only and first order only (with the exception of enforcing $f = 0$ at $x_1 = 0$, $x_2 = 0$) surrogate models are depicted in Figures 7.7 (a) and (b). In addition, the first order partial derivatives w.r.t. x_1 and x_2 of the surrogate models are respectively depicted in Figures 7.8 (a) and (b) and Figures 7.8 (c) and (d) over the domain -5 to 5 for both variables. The approximated associated partial derivatives of the zero order only surrogate model differs significantly from the actual associated partial derivatives depicted in Figures 7.4 (c) and (d). However, the associated partial derivatives of the first order only approximated surrogate model is nearly identical to the actual associated partial derivatives. Similarly, the zero order only constructed surrogate model

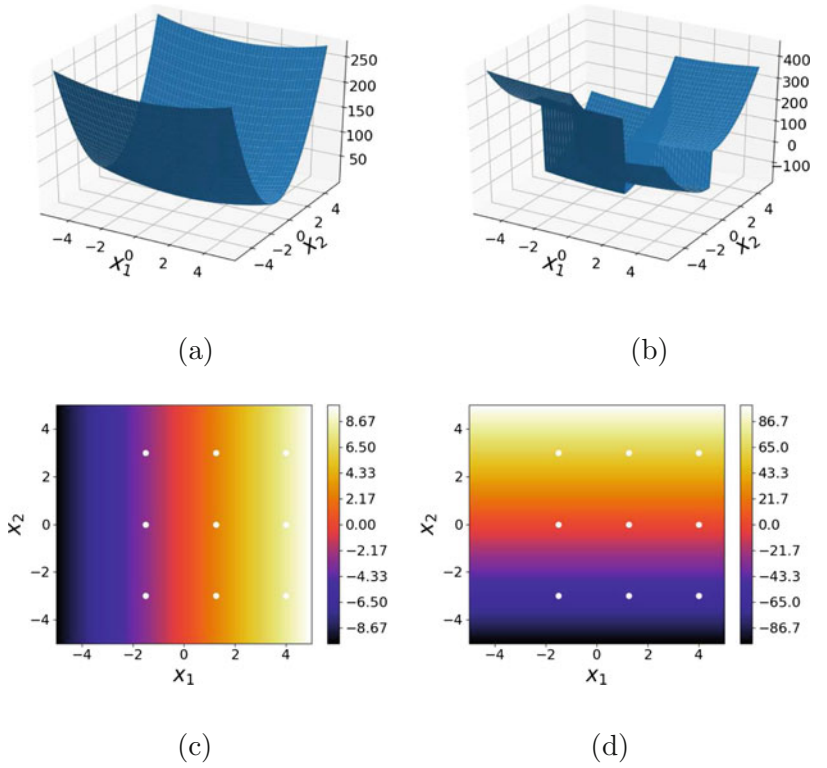


Figure 7.4: Surface of (a) the smooth quadratic function given by (7.13), and (b) the piece-wise smooth step discontinuous quadratic function given by (7.14). In addition, the first associated partial derivatives w.r.t. x_1 and x_2 for both functions are the same and respectively depicted in (c) and (d), with the full set of identified designs points indicated by white circles

differs significantly from both the actual smooth quadratic function and the piece-wise smooth step discontinuous function respectively depicted in Figures 7.4 (a) and (b). In contrast, the first order only constructed surrogate model is nearly identical to the actual smooth quadratic function, demonstrating that the step discontinuities are effectively ignored when only first order information is considered.

It is evident that the zero order only and first order only constructed surrogate functions differ significantly in approximating both the func-

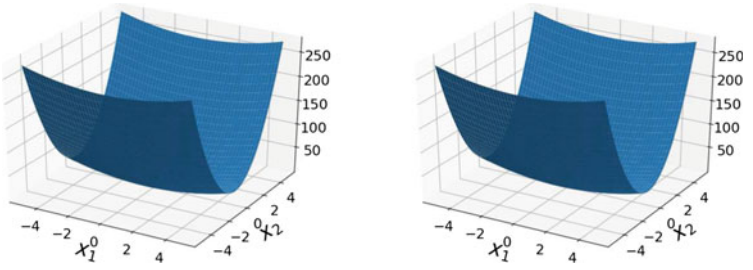


Figure 7.5: (a) Zero order only and (b) first order only constructed response surfaces for the smooth quadratic function

tion value and associated gradient of the actual piece-wise smooth step discontinuous function. The zero order constructed surrogate function poorly represents both the function and associated gradients of the piece-wise smooth step discontinuous function. It is only the approximated associated gradients of the first order constructed surrogate function that is consistent with the actual associated gradient of the piece-wise smooth step discontinuous function. The result, when only first order information is considered to construct a surrogate, is a smooth surrogate that is consistent with the first order information of the piece-wise smooth step discontinuous function. As will be pointed out in Chapter 8, this is ideal when the step discontinuities are numerical artefacts that need to be ignored. The effectiveness of first order only constructed surrogates to ignore or filter out step discontinuities is remarkable and an important aspect to consider when constructing surrogates for discontinuous functions. The resulting smooth surrogate function that is approximated from only first order information can then be optimized using conventional gradient based approaches.

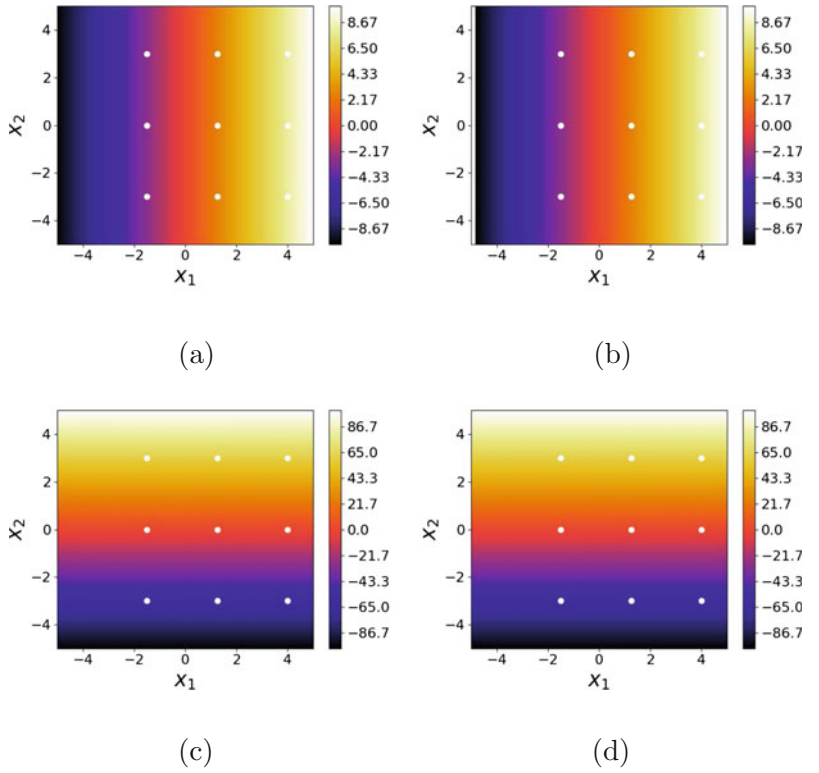


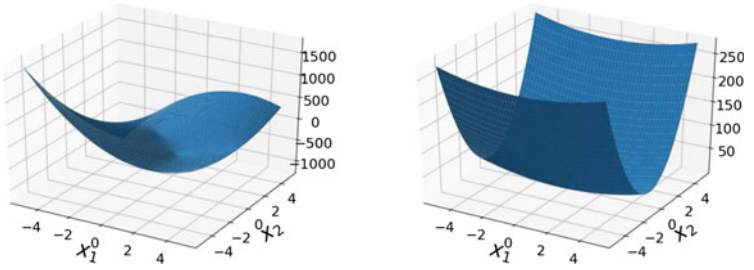
Figure 7.6: First order partial derivatives w.r.t. x_1 and x_2 respectively computed from the constructed (a),(c) zero order only surrogate model and (b),(d) first order only surrogate model for the smooth quadratic function. The sampled designs are indicated by white circles

7.5 Exercises

The reader is encouraged to employ a convenient computing environment to complete the exercises. With Python being freely available it is recommended to be used as outlined in Chapter 9.

7.4.1 Consider some non-linear function $f(\mathbf{x}, \mathbf{c})$ that is linear w.r.t. some parametrization $\mathbf{c} = [c_0, c_1, \ldots, c_r]$. For example consider the quadratic function

$$f(\mathbf{x}, \mathbf{c}) = c_5 x_1^2 + c_4 x_2^2 + c_3 x_1 x_2 + c_2 x_1 + c_1 x_2 + c_0,$$



(a) (b)

Figure 7.7: (a) Zero order only and (b) first order only constructed response surfaces for the piece-wise smooth step discontinuous quadratic function

that is non-linear in $\mathbf{x}$ but linear in $\mathbf{c}$. For unknown $\mathbf{c} = \mathbf{c}^*$, given k observations $f(\mathbf{x}^0, \mathbf{c}^*), f(\mathbf{x}^1, \mathbf{c}^*) \ldots, f(\mathbf{x}^k, \mathbf{c}^*)$ for known $\mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^k$ and with $k > r$. Formulate an unconstrained minimization problem, using matrix notation, that estimates $\mathbf{c}^*$ from the information of k observations.

7.4.2 Derive the general first order optimality criterion that solves for $\mathbf{c}^*$ for the formulation presented in Problem **7.4.1**.

7.4.3 Equation 7.15 represents 12 scalar observations $f^i = f(\mathbf{x}^i, \mathbf{c}^*),\ i = 1, \ldots, 12$ that depends on two variables x_1 and x_2.

i	x_1^i	x_2^i	f^i
1	−3	−3	9
2	−1.5	−3	8
3	0	−3	7
4	1.5	−3	6
5	−3	0	0.5
6	−1.5	0	1.0
7	0	0	1.25
8	1.5	0	1.5
9	−3	3	6
10	−1.5	3	7
11	0	3	8
12	1.5	3	9

(7.15)

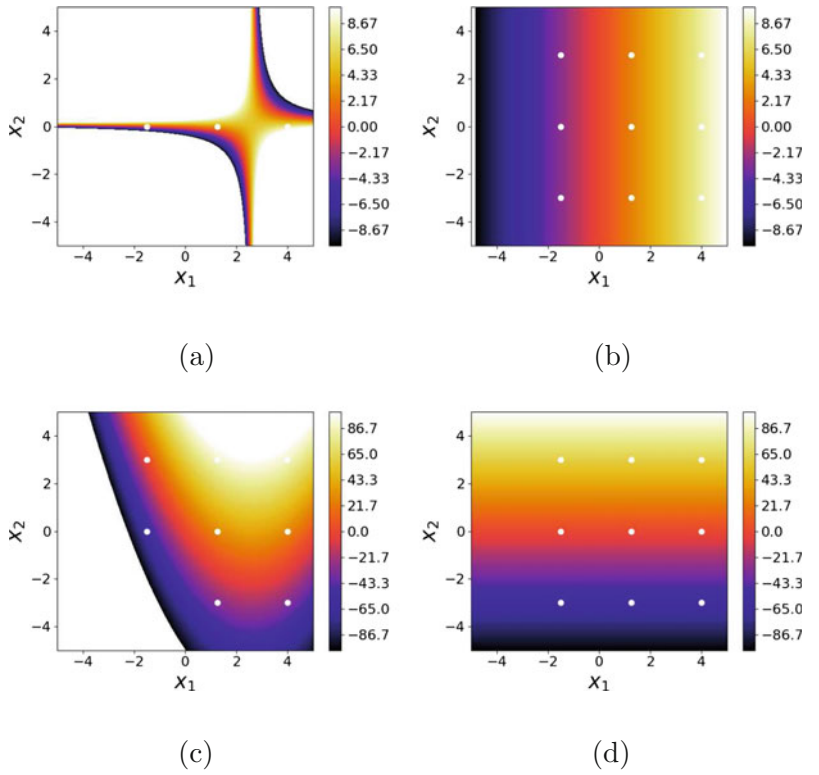


Figure 7.8: First order partial derivatives w.r.t. x_1 and x_2 respectively computed from the constructed (a),(c) zero order only surrogate model and (b),(d) first order only surrogate model for the piece-wise smooth step discontinuous quadratic function. The sampled designs are indicated by white circles and the colorbar scale limited to the minimum and maximum values of the actual associated partial derivatives

Fit the following quadratic function

$$f(\mathbf{x}, \mathbf{c}) = c_5 x_1^2 + c_4 x_2^2 + c_3 x_1 x_2 + c_2 x_1 + c_1 x_2 + c_0, \qquad (7.16)$$

to the data in Equation 7.15 using the optimal criterion derived in Exercise 7.4.2.

7.4.4 For Exercise 7.4.3 present, in a detailed discussion, all the verifications you can envisage to ensure that it was indeed correctly solved.

7.4.5 Verify the solution in Exercise 7.4.3 by outlining and conducting a numerical study that perturbs the optimal solution.

7.4.6 Verify the solution in Exercise 7.4.3 by outlining and conducting a numerical study that solves the formulated problem iteratively using an appropriate minimization algorithm.

7.4.7 Instead of using the f^i given in Equation 7.15, evaluate the function $f(\mathbf{x}) = 13x_1^2 + 11x_2^2 + 7x_1x_2 + 5x_1 + 3$ at the designs listed in Equation 7.15. What do you expect the solution for $c_0, c_1, \ldots, c_5$ to be, and motivate your answer by a detailed discussion of your reasoning.

7.4.8 Conduct a numerical study that investigates the validity of your expectation in Exercise 7.4.7.

7.4.9 Higher order polynomials compute higher order powers of designs of $\mathbf{x}$ that may lead to numerically very large numbers and ultimately to ill-conditioning of the system to be solved. Propose an appropriate scaling of $\mathbf{x}$ that would eliminate such ill-conditioning.

7.4.10 Given two Gaussian representations, $\phi_1(r) = e^{-\epsilon r^2}$ and $\phi_2(r) = e^{-\frac{r^2}{\epsilon}}$. Discuss the implication of increasing ϵ on the size of the effect of the domain covered by $\phi_1(r)$ and $\phi_2(r)$ respectively.

7.4.11 Utilize the function `Rbf` in the module `scipy.interpolate` to construct a zero order only radial basis surrogate model of the quadratic function in Exercise 7.4.7 using the Gaussian basis function. Note that `Rbf` returns a function object that takes the same number of inputs as the dimensionality of the problem, where each entry is a list of values for a coordinate at which to evaluate the constructed RBF function. The basis function can be selected by assigning the string `gaussian` to the parameter `function` of the function object, and the shape parameter by assigning a value to the parameter `epsilon`. Type `help(Rbf)` for additional information and note the usage of the shape parameter ϵ in the Gaussian formulation.

7.4.12 Optimize the actual quadratic function given in Exercise 7.4.7 using 100 random starting guesses between -1.5 and 1.5 for both x_1 and x_2 using a zero and first order algorithm of your choice.

Repeat the optimization using the same initial starting guesses, but this time optimize the RBF approximation to the quadratic function constructed in Exercise 7.4.11. Critically compare the solutions obtained.

7.4.13 Repeat Exercise 7.4.12 but increase the bounds for the initial starting guesses from -1.5 and 1.5 to (i) -15 and 15, and to (ii) -150 and 150 respectively and critically compare the solutions obtained as well as pointing out the cause of any evident changes in the results.

7.4.14 Repeat Exercise 7.4.12 but use the Exponential basis function given by $\phi(r) = e^{-\epsilon r}$ (Python code `exp(-(r*self.epsilon))`). Critically compare the solutions obtained as well as pointing out the cause of any evident changes in the results.

7.4.15 Given the Gaussian radial basis function $\phi(r(\mathbf{x})) = e^{-\epsilon r(\mathbf{x})^2)}$, and the Exponential radial basis function $\phi(r(\mathbf{x})) = e^{-\epsilon r(\mathbf{x})}$. Plot the functions and critically discuss any expected implications of the smoothness of the basis function on the performance of gradient based optimization strategies. Which basis function would you prefer for gradient based optimization strategies?

7.4.16 Consider the Gaussian basis function given in (7.11). Conduct a numerical study that identifies the observed ill-conditioning of $\mathbf{R}_{zo}$ as ϵ gets smaller, and the absence of ill-conditioning as ϵ gets larger.

7.4.17 Given a two-dimensional function defined over the domain -1 and 1 for x_1, and -100 and 100 for x_2. Discuss the potential problems associated with constructing an RBF surrogate model over such a domain. Detail a potential solution strategy.

7.4.18 Conduct a numerical investigation that highlights the potential difficulties associated with Exercise 7.4.17 and that demonstrates the potential benefits of the solution strategy proposed in Exercise 7.4.17.

7.4.19 Write your own Python code to construct a zero order only radial basis surrogate model for a function of n dimensions. Test your code on the quadratic function given in Exercise 7.4.7, and

compare your RBF approximation against `scipy.interpolate.Rbf` for equivalent shape parameters.

7.4.20 Approximate the gradient vector of the quadratic function given in Exercise 7.4.7. Use the radial basis surrogate model constructed in Exercise 7.4.19 at 100 random designs over the domain defined by the design of experiments. Compute the average error for each component of the gradient vector at the 100 random designs.

7.4.21 Write your own Python code to construct a combined zero and first order radial basis surrogate model for a function of n dimensions. Test your code on the quadratic function given in Exercise 7.4.7.