Chapter 3
Surrogate-Based Methods

Slawomir Koziel, David Echeverría Ciaurri, and Leifur Leifsson

Abstract. Objective functions that appear in engineering practice may come from measurements of physical systems and, more often, from computer simulations. In many cases, optimization of such objectives in a straightforward way, i.e., by applying optimization routines directly to these functions, is impractical. One reason is that simulation-based objective functions are often analytically intractable (discontinuous, non-differentiable, and inherently noisy). Also, sensitivity information is usually unavailable, or too expensive to compute. Another, and in many cases even more important, reason is the high computational cost of measurement/simulations. Simulation times of several hours, days or even weeks per objective function evaluation are not uncommon in contemporary engineering, despite the increase of available computing power. Feasible handling of these unmanageable functions can be accomplished using surrogate models: the optimization of the original objective is replaced by iterative re-optimization and updating of the analytically tractable and computationally cheap surrogate. This chapter briefly describes the basics of surrogate-based optimization, various ways of creating surrogate models, as well as several examples of surrogate-based optimization techniques.

Keywords: Surrogate-based optimization, multi-fidelity optimization, surrogate models, simulation-driven design, trust-region methods, function approximation, design of experiments.

Slawomir Koziel · Leifur Leifsson
Engineering Optimization & Modeling Center, School of Science and Engineering,
Reykjavik University, Menntavegur 1, 101 Reykjavik, Iceland
email: {koziel,leifurth}@ru.is

David Echeverría Ciaurri
Department of Energy Resources Engineering,
Stanford University, Stanford, CA 94305-2220, USA
email: echeverr@stanford.edu

S. Koziel & X.-S. Yang (Eds.): Comput. Optimization, Methods and Algorithms, SCI 356, pp. 33–59.
springerlink.com

3.1 Introduction

Contemporary engineering is more and more dependent on computer-aided design (CAD). In most engineering fields, numerical simulations are used extensively, not only for design verification but also directly in the design process. As a matter of fact, because of increasing system complexity, ready-to-use theoretical (e.g., analytical) models are not available in many cases. Thus, simulation-driven design and design optimization becomes the only option to meet the specifications prescribed, improve the system reliability, or reduce the fabrication cost.

The simulation-driven design can be formulated as a nonlinear minimization problem of the following form

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}), \tag{3.1}$$

where $f(\boldsymbol{x})$ denotes the objective function to be minimized evaluated at the point $\boldsymbol{x} \in R^n$ ($\boldsymbol{x}$ is the design variable vector). In many engineering problems f is of the form $f(\boldsymbol{x}) = U(\boldsymbol{R}_f(\boldsymbol{x}))$, where $\boldsymbol{R}_f \in R^m$ denotes the response vector of the system of interest (in particular, one may have $m > n$ or even $m >> n$ [1]), whereas U is a given scalar merit function. In particular, U can be defined through a norm that measures the distance between $\boldsymbol{R}_f(\boldsymbol{x})$ and a target vector $\boldsymbol{y}$. An optimal design vector is denoted by $\boldsymbol{x}^*$. In many cases, $\boldsymbol{R}_f$ is obtained through computationally expensive computer simulations. We will refer to it as a high-fidelity or fine model. To simplify notation, f itself will also be referred to as the high-fidelity (fine) model.

Unfortunately, a direct attempt to solve (3.1) by embedding the simulator directly in the optimization loop may be impractical. The underlying simulations can be very time-consuming (in some instances, the simulation time can be as long as several hours, days or even weeks per single design), and the presence of massive computing resources is not always translated in computational speedup. This latter fact is due to a growing demand for simulation accuracy, both by including multiphysics and second-order effects, and by using finer discretization of the structure under consideration. As conventional optimization algorithms (e.g., gradient-based schemes with numerical derivatives) require tens, hundreds or even thousands of objective function calls per run (that depends on the number of design variables), the computational cost of the whole optimization process may not be acceptable.

Another problem is that objective functions coming from computer simulations are often analytically intractable (i.e., discontinuous, non-differentiable, and inherently noisy). Moreover, sensitivity information is frequently unavailable, or too expensive to compute. While in some cases it is possible to obtain derivative information inexpensively through adjoint sensitivities [2], numerical noise is an important issue that can complicate simulation-driven design. We should also mention that adjoint-based sensitivities require detailed knowledge of and access to the simulator source code, and this is something that cannot be assumed to be generally available.

Surrogate-based optimization (SBO) [1,3,4] has been suggested as an effective approach for the design with time-consuming computer models. The basic concept

of SBO is that the direct optimization of the computationally expensive model is replaced by an iterative process that involves the creation, optimization and updating of a fast and analytically tractable surrogate model. The surrogate should be a reasonably accurate representation of the high-fidelity model, at least locally. The design obtained through optimizing the surrogate model is verified by evaluating the high-fidelity model. The high-fidelity model data obtained in this verification process is then used to update the surrogate. SBO proceeds in this predictor-corrector fashion iteratively until some termination criterion is met. Because most of the operations are performed on the surrogate model, SBO reduces the computational cost of the optimization process when compared to optimizing the high-fidelity model directly, without resorting to any surrogate.

In this chapter, we review the basics of surrogate-based optimization. We briefly present various ways of generating surrogate models, and we emphasize on the distinction between models based on function approximations of sampled high-fidelity model data and models constructed from physically-based low-fidelity models. A few selected surrogate-based optimization algorithms including space mapping [1,5,6], approximation model management [7], manifold mapping [8], and the surrogate-management framework [9], are also discussed. We conclude the chapter with some final remarks.

3.2 Surrogate-Based Optimization

As mentioned in the introduction, there are several reasons why the straightforward optimization of the high-fidelity model may not work or can be impractical. These reasons include high computational cost of each model evaluation, numerical noise and discontinuities in the cost function. Surrogate-based optimization [3,5] aims at alleviating such problems by using an auxiliary model, the surrogate, that is preferably fast, amenable to optimization, and yet reasonably accurate. One popular approach for constructing surrogate models is through approximations of high-fidelity model data obtained by sampling the design space using appropriate design of experiments methodologies [3]. Some of these strategies for allocating samples [10], generating approximations [3,4,10], as well as validating the surrogates are discussed in Section 3.3.

The surrogate model optimization yields an approximation of the minimizer associated to the high-fidelity model. This approximation has to be verified by evaluating the high-fidelity model at the predicted high-fidelity model minimizer. Depending on the result of this verification, the optimization process may be terminated. Otherwise, the surrogate model is updated using the new available high-fidelity model data, and then re-optimized to obtain a new, and hopefully better, approximation of the high-fidelity model minimizer.

The surrogate-based optimization process can be summarized as follows (Fig. 3.1):

1. Generate the initial surrogate model.
2. Obtain an approximate solution to (3.1) by optimizing the surrogate.

3. Evaluate the high-fidelity model at the approximate solution computed in Step 2.
4. Update the surrogate model using the new high-fidelity model data.
5. Stop if the termination condition is satisfied; otherwise go to Step 2.

The SBO framework can be formulated as an iterative procedure [3,5]:

$$\boldsymbol{x}^{(i+1)} = \arg\min_{\boldsymbol{x}} s^{(i)}(\boldsymbol{x}) \cdot \tag{3.2}$$

This scheme generates a sequence of points (designs) $\boldsymbol{x}^{(i)}$ that (hopefully) converge to a solution (or a good approximation) of the original design problem (3.1). Each $\boldsymbol{x}^{(i+1)}$ is the optimal design of the surrogate model $s^{(i)}$, which is assumed to be a computationally cheap and sufficiently reliable representation of the fine model f, particularly in the neighborhood of the current design $\boldsymbol{x}^{(i)}$. Under these assumptions, the algorithm (3.2) aims at a sequence of designs to quickly approach $\boldsymbol{x}^*$. Typically, and for verification purposes, the high-fidelity model is evaluated only once per iteration (at every new design $\boldsymbol{x}^{(i+1)}$). The data obtained from the validation is used to update the surrogate model. Because the surrogate model is computationally cheap, the optimization cost associated with (3.2) can—in many cases—be viewed as negligible, so that the total optimization cost is determined by the evaluation of the high-fidelity model. Normally, the number of iterations often needed within a surrogate-based optimization algorithm is substantially smaller than for any method that optimizes the high-fidelity model directly (e.g., gradient-based schemes with numerical derivatives) [5].

If the surrogate model satisfies zero- and first-order consistency conditions with the high-fidelity model (i.e., $s^{(i)}(\boldsymbol{x}^{(i)}) = f(\boldsymbol{x}^{(i)})$ and $\nabla s^{(i)}(\boldsymbol{x}^{(i)}) = \nabla f(\boldsymbol{x}^{(i)})$ [7]; it should be noticed that the verification of the latter requires high-fidelity model sensitivity data), and the surrogate-based algorithm is enhanced by, for example, a trust region method [11] (see Section 3.4.1), then the sequence of intermediate solutions is provably convergent to a local optimizer of the fine model [12] (some standard assumptions concerning the smoothness of the functions involved are also necessary) [13]. Convergence can also be guaranteed if the SBO algorithm is embedded within the framework given in [5,14] (space mapping), [13] (manifold mapping) or [9] (surrogate management framework). A more detailed description of several surrogate-based optimization techniques is given in Section 3.4.

Space mapping [1,5,6] is an example of a surrogate-based methodology that does not normally rely on using sensitivity data or trust region convergence safeguards; however, it requires the surrogate model to be constructed from a physically-based coarse model [1]. This usually gives remarkably good performance in the sense of the algorithm being able to locate a satisfactory design quickly. Unfortunately space mapping suffers from convergence problems [14] and it is sensitive to the quality of the coarse model and the specific analytical formulation of the surrogate [15,16].

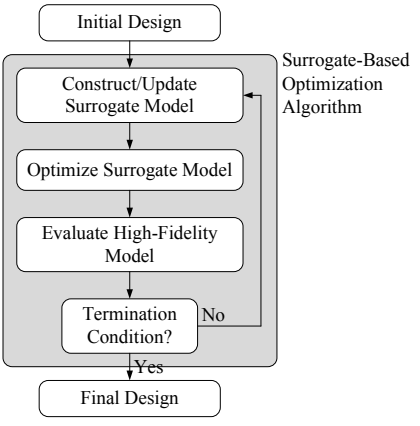


Fig. 3.1 Flowchart of the surrogate-based optimization process. An approximate high-fidelity model minimizer is obtained iteratively by optimizing the surrogate model. The high-fidelity model is evaluated at each new design for verification purposes. If the termination condition is not satisfied, the surrogate model is updated and the search continues. In most cases the high-fidelity model is evaluated only once per iteration. The number of iterations needed in SBO is often substantially smaller than for conventional (direct) optimization techniques.

3.3 Surrogate Models

The surrogate model is a key component of any SBO algorithm. It has to be computationally cheap, preferably smooth, and, at the same time, reasonably accurate, so that it can be used to predict the location of high-fidelity model minimizers. We can clearly distinguish between physical and functional surrogate models.

Physical (or physically-based) surrogates are constructed from an underlying low-fidelity (coarse) model. The low-fidelity model is a representation of the system of interest with relaxed accuracy [1]. Coarse models are computationally cheaper than high-fidelity models and, in many cases, have better analytical properties. The low-fidelity model can be obtained, for example, from the same simulator as the one used for the high-fidelity model but using a coarse discretization [17]. Alternatively, the low-fidelity model can be based on simplified physics (e.g., by exploiting simplified equations [1], or by neglecting certain second-order effects) [18], or on a significantly different physical description (e.g., lumped parameter versus partial differential equation based models [1]). In some cases, low-fidelity models can be formulated using analytical or semi-empirical formulas [19]. The coarse model can be corrected if additional data from the high-fidelity model is available (for example, during the course of the optimization).

In general, physical surrogate models are:

- based on particular knowledge about the physical system of interest,
- dedicated (reuse across different designs is uncommon),
- more expensive to evaluate and more accurate (in a global sense) than functional surrogates.

It should be noticed that the evaluation of a physical surrogate may involve, for example, the numerical solution of partial differential equations or even actual measurements of the physical system.

The main advantage of physically-based surrogates is that the amount of high-fidelity model data necessary for obtaining a given level of accuracy is generally substantially smaller than for functional surrogates (physical surrogates inherently embed knowledge about the system of interest) [1]. Hence, surrogate-based optimization algorithms that exploit physically-based surrogate models are usually more efficient than those using functional surrogates (in terms of the number of high-fidelity model evaluations required to find a satisfactory design) [5].

Functional (or approximation) surrogate models [20,4]:

- can be constructed without previous knowledge of the physical system of interest,
- are generic, and therefore applicable to a wide class of problems,
- are based on (usually simple) algebraic models,
- are often very cheap to evaluate but require considerable amount of data to ensure reasonable general accuracy.

An initial functional surrogate can be generated using high-fidelity model data obtained through sampling of the design space. Figure 3.2 shows the model construction flowchart for a functional surrogate. Design of experiments involves the use of strategies for allocating samples within the design space. The particular choice depends on the number of samples one can afford (in some occasions only a few points may be allowed), but also on the specific modeling technique that will be used to create the surrogate. Though in some cases the surrogate can be found using explicit formulas (e.g., polynomial approximation) [3], in most situations it is computed by means of a separate minimization problem (e.g., when using kriging [21] or neural networks [22]). The accuracy of the model should be tested in order to estimate its prediction/generalization capability. The main difficulty in obtaining a good functional surrogate lies in keeping a balance between accuracy at the known and at the unknown data (training and testing set, respectively). The surrogate could be subsequently updated using new high-fidelity model data that is accumulated during the run of the surrogate-based optimization algorithm.

In this section we first describe the fundamental steps for generating functional surrogates. Various sampling techniques are presented in Section 3.3.1. The surrogate creation and the model validations steps are tackled in Section 3.3.2 and Section 3.3.3, respectively. If the quality of the surrogate is not sufficient, more data points can be added, and/or the model parameters can be updated to improve accuracy. Several correction methods, both for functional and physical surrogates, are described in Section 3.3.4.

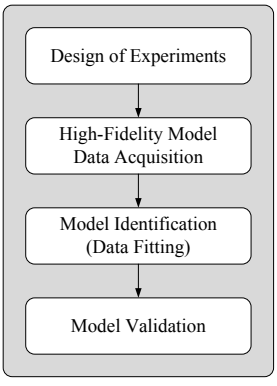


Fig. 3.2 Surrogate model construction flowchart. If the quality of the model is not satisfactory, the procedure can be iterated (more data points will be required).

3.3.1 Design of Experiments

Design of experiments (DOE) [23,24,25] is a strategy for allocating samples (points) in the design space that aims at maximizing the amount of information acquired. The high-fidelity model is evaluated at these points to create the training data set that is subsequently used to construct the functional surrogate model. When sampling, there is a clear trade-off between the number of points used and the amount of information that can be extracted from these points. The samples are typically spread apart as much as possible in order to capture global trends in the design space.

Factorial designs [23] are classical DOE techniques that, when applied to discrete design variables, explore a large region of the search space. The sampling of all possible combinations is called full factorial design. Fractional factorial designs can be used when model evaluation is expensive and the number of design variables is large (in full factorial design the number of samples increases exponentially with the number of design variables). Continuous variables, once discretized, can be easily analyzed through factorial design. Full factorial two-level and three-level design (also known as 2^k and 3^k design) allows us to estimate main effects and interactions between design variables, and quadratic effects and interactions, respectively. Figures 3.3(a) and 3.3(b) show examples of full two-level and fractional two-level design, respectively, for three design variables (i.e., $n = 3$). Alternative factorial designs can be found in practice: central composite design (see Figure 3.3(c)), star design (frequently used in combination with space mapping [26]; see Figure 3.3(d)), or Box-Behnken design (see Figure 3.3(e)).

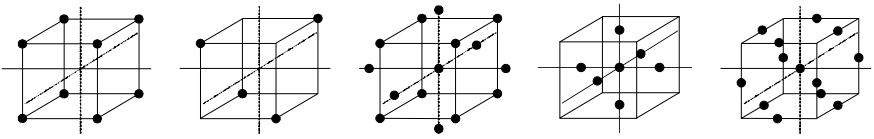


Fig. 3.3 Factorial designs for three design variables (n = 3): (a) full factorial design, (b) fractional factorial design, (c) central composite design, (d) star design, and (e) Box-Behnken design.

If no prior knowledge about the objective function is available (typical while constructing the initial surrogate), some recent DOE approaches tend to allocate the samples uniformly within the design space [3]. A variety of space filling designs are available. The simplest ones do not ensure sufficient uniformity (e.g., pseudo-random sampling [23]) or are not practical (e.g., stratified random sampling, where the number of samples needed is on the order of 2^n). One of the most popular DOE for (relatively) uniform sample distributions is Latin hypercube sampling (LHS) [27]. In order to allocate p samples with LHS, the range for each parameter is divided into p bins, which for n design variables, yields a total number of p^n bins in the design space. The samples are randomly selected in the design space so that (i) each sample is randomly placed inside a bin, and (ii) for all one-dimensional projections of the p samples and bins, there is exactly one sample in each bin. Figure 3.4 shows a LHS realization of 15 samples for two design variables (n = 2). It should be noted that the standard LHS may lead to non-uniform distributions (for example, samples allocated along the design space diagonal satisfy conditions (i) and (ii)). Numerous improvements of standard LHS, e.g., [28]-[31], provide more uniform sampling distributions.

Other DOE methodologies commonly used include orthogonal array sampling [3], quasi-Monte Carlo sampling [23], or Hammersley sampling [23]. Sample distribution can be improved through the incorporation of optimization techniques that minimize a specific non-uniformity measure, e.g., $\sum_{i=1}^{p}\sum_{j=i+1}^{p} d_{ij}^{-2}$ [29], where d_{ij} is the Euclidean distance between samples i and j.

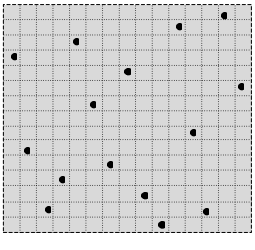


Fig. 3.4 Latin hypercube sampling realization of 15 samples in a two-dimensional design space.

3.3.2 Surrogate Modeling Techniques

Having selected the design of experiments technique and sampled the data, the next step is to choose an approximation model and a fitting methodology. In this section, we describe in some detail the most popular surrogate modeling techniques, and we briefly mention alternatives.

3.3.2.1 Polynomial Regression

Polynomial regression [3] assumes the following relation between the function of interest f and K polynomial basis functions v_j using p samples $f(\boldsymbol{x}^{(i)})$, $i = 1, \ldots, p$:

$$f(\boldsymbol{x}^{(i)}) = \sum_{j=1}^{K} \beta_j v_j(\boldsymbol{x}^{(i)}) \cdot \tag{3.3}$$

These equations can be represented in matrix form

$$\boldsymbol{f} = \boldsymbol{X}\boldsymbol{\beta}, \tag{3.4}$$

where $\boldsymbol{f} = [f(\boldsymbol{x}^{(1)})\ f(\boldsymbol{x}^{(2)})\ \ldots\ f(\boldsymbol{x}^{(p)})]^T$, $\boldsymbol{X}$ is a $p \times K$ matrix containing the basis functions evaluated at the sample points, and $\boldsymbol{\beta} = [\beta_1\ \beta_2\ \ldots\ \beta_K]^T$. The number of sample points p should be consistent with the number of basis functions considered K (typically $p \geq K$). If the sample points and basis function are taken arbitrarily, some columns of $\boldsymbol{X}$ can be linearly dependent. If $p \geq K$ and rank($\boldsymbol{X}$) = K, a solution of (3.4) in the least-squares sense can be computed through $\boldsymbol{X}^+$, the pseudoinverse (or generalized inverse) of $\boldsymbol{X}$ [32]:

$$\boldsymbol{\beta} = \boldsymbol{X}^+ = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{f}. \tag{3.5}$$

The simplest examples of regression models are the first- and second-order order polynomial models

$$s(\boldsymbol{x}) = s([x_1\ x_2\ \ldots\ x_n]^T) = \beta_0 + \sum_{j=1}^{n} \beta_j x_j\ , \tag{3.6}$$

$$s(\boldsymbol{x}) = s([x_1\ x_2\ \ldots\ x_n]^T) = \beta_0 + \sum_{j=1}^{n} \beta_j x_j + \sum_{i=1}^{n}\sum_{j\leq i}^{n} \beta_{ij} x_i x_j \cdot \tag{3.7}$$

Polynomial interpolation/regression appears naturally and is crucial in developing robust and efficient derivative-free optimization algorithms. For more details, please refer to [33].

3.3.2.2 Radial Basis Functions

Radial basis function interpolation/approximation [4,34] exploits linear combinations of K radially symmetric functions ϕ

$$s(\boldsymbol{x}) = \sum_{j=1}^{K} \lambda_j \phi(\| \boldsymbol{x} - \boldsymbol{c}^{(j)} \|), \tag{3.8}$$

where $\boldsymbol{\lambda} = [\lambda_1\ \lambda_2\ \ldots\ \lambda_K]^T$ is the vector of model parameters, and $\boldsymbol{c}^{(j)}$, $j = 1, \ldots, K$, are the (known) basis function centers.

As in polynomial regression the model parameters $\boldsymbol{\lambda}$ can be computed by

$$\boldsymbol{\lambda} = \boldsymbol{\Phi}^{+} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{f}\,, \tag{3.9}$$

where again $\boldsymbol{f} = [f(\boldsymbol{x}^{(1)})\ f(\boldsymbol{x}^{(2)})\ \ldots\ f(\boldsymbol{x}^{(p)})]^T$, and the $p\times K$ matrix $\boldsymbol{\Phi}$ is defined as

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi(\|\boldsymbol{x}^{(1)} - \boldsymbol{c}^{(1)}\|) & \phi(\|\boldsymbol{x}^{(1)} - \boldsymbol{c}^{(2)}\|) & \cdots & \phi(\|\boldsymbol{x}^{(1)} - \boldsymbol{c}^{(K)}\|) \\ \phi(\|\boldsymbol{x}^{(2)} - \boldsymbol{c}^{(1)}\|) & \phi(\|\boldsymbol{x}^{(2)} - \boldsymbol{c}^{(2)}\|) & \cdots & \phi(\|\boldsymbol{x}^{(2)} - \boldsymbol{c}^{(K)}\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|\boldsymbol{x}^{(p)} - \boldsymbol{c}^{(1)}\|) & \phi(\|\boldsymbol{x}^{(p)} - \boldsymbol{c}^{(2)}\|) & \cdots & \phi(\|\boldsymbol{x}^{(p)} - \boldsymbol{c}^{(K)}\|) \end{bmatrix}. \tag{3.10}$$

If we select $p = K$ (i.e., the number of basis functions is equal to the number of samples), and if the centers of the basis functions coincide with the data points (and these are all different), $\boldsymbol{\Phi}$ is a regular square matrix (and thus, $\boldsymbol{\lambda} = \boldsymbol{\Phi}^{-1}\boldsymbol{f}$).

Typical choices for the basis functions are $\phi(r) = r$, $\phi(r) = r^3$, or $\phi(r) = r^2\ln r$ (thin plate spline). More flexibility can be obtained by using parametric basis functions such as $\phi(r) = \exp(-r^2/2\sigma^2)$ (Gaussian), $\phi(r) = (r^2 + \sigma^2)^{1/2}$ (multi-quadric), or $\phi(r) = (r^2 + \sigma^2)^{-1/2}$ (inverse multi-quadric).

3.3.2.3 Kriging

Kriging is a popular technique to interpolate deterministic noise-free data [35,10,21,36]. Kriging is a Gaussian process [37] based modeling method, which is compact and cheap to evaluate. Kriging has been proven to be useful in a wide variety of fields (see, e.g., [4,38] for applications in optimization).

In its basic formulation, kriging [35,10] assumes that the function of interest f is of the form

$$f(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{\beta} + Z(\boldsymbol{x})\,, \tag{3.11}$$

where $\boldsymbol{g}(\boldsymbol{x}) = [g_1(\boldsymbol{x})\ g_2(\boldsymbol{x})\ \ldots\ g_K(\boldsymbol{x})]^T$ are known (e.g., constant) functions, $\boldsymbol{\beta} = [\beta_1\ \beta_2\ \ldots\ \beta_K]^T$ are the unknown model parameters, and $Z(\boldsymbol{x})$ is a realization of a normally distributed Gaussian random process with zero mean and variance σ^2.

The regression part $\boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{\beta}$ approximates globally the function f, and $Z(\boldsymbol{x})$ takes into account localized variations. The covariance matrix of $Z(\boldsymbol{x})$ is given as

$$Cov[Z(\boldsymbol{x}^{(i)})Z(\boldsymbol{x}^{(j)})] = \sigma^2\boldsymbol{R}([R(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})])\,, \tag{3.12}$$

where $\boldsymbol{R}$ is a $p\times p$ correlation matrix with $R_{ij} = R(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)})$. Here, $R(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$ is the correlation function between sampled data points $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)}$. The most popular choice is the Gaussian correlation function

$$R(\boldsymbol{x}, \boldsymbol{y}) = \exp\left[-\sum_{k=1}^{n}\theta_k\,|\,x_k - y_k\,|^2\right]\,, \tag{3.13}$$

where θ_k are unknown correlation parameters, and x_k and y_k are the k^{th} component of the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively.

The kriging predictor [10,35] is defined as

$$s(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x})^T \boldsymbol{\beta} + \boldsymbol{r}^T(\boldsymbol{x}) \boldsymbol{R}^{-1} (\boldsymbol{f} - \boldsymbol{G}\boldsymbol{\beta}), \tag{3.14}$$

where $\boldsymbol{r}(\boldsymbol{x}) = [R(\boldsymbol{x}, \boldsymbol{x}^{(1)}) \ \ldots \ R(\boldsymbol{x}, \boldsymbol{x}^{(p)})]^T$, $\boldsymbol{f} = [f(\boldsymbol{x}^{(1)}) \ f(\boldsymbol{x}^{(2)}) \ \ldots \ f(\boldsymbol{x}^{(p)})]^T$, and $\boldsymbol{G}$ is a $p \times K$ matrix with $G_{ij} = g_j(\boldsymbol{x}^{(i)})$.

The vector of model parameters $\boldsymbol{\beta}$ can be computed as

$$\boldsymbol{\beta} = (\boldsymbol{G}^T \boldsymbol{R}^{-1} \boldsymbol{G})^{-1} \boldsymbol{G}^T \boldsymbol{R}^{-1} \boldsymbol{f} \ . \tag{3.15}$$

An estimate of the variance σ^2 is given by

$$\hat{\sigma}^2 = \frac{1}{p} (\boldsymbol{f} - \boldsymbol{G}\boldsymbol{\beta})^T \boldsymbol{R}^{-1} (\boldsymbol{f} - \boldsymbol{G}\boldsymbol{\beta}) \cdot \tag{3.16}$$

Model fitting is accomplished by maximum likelihood for θ_k [35]. In particular, the n-dimensional unconstrained nonlinear maximization problem with cost function

$$-(p \ln(\hat{\sigma}^2) + \ln |\boldsymbol{R}|)/2, \tag{3.17}$$

where the variance σ^2 and $|\boldsymbol{R}|$ are both functions of θ_k, is solved for positive values of θ_k as optimization variables.

It should be noted that, once the kriging-based surrogate has been obtained, the random process $Z(\boldsymbol{x})$ gives valuable information regarding the approximation error that can be used for improving the surrogate [4,35].

3.3.2.4 Neural Networks

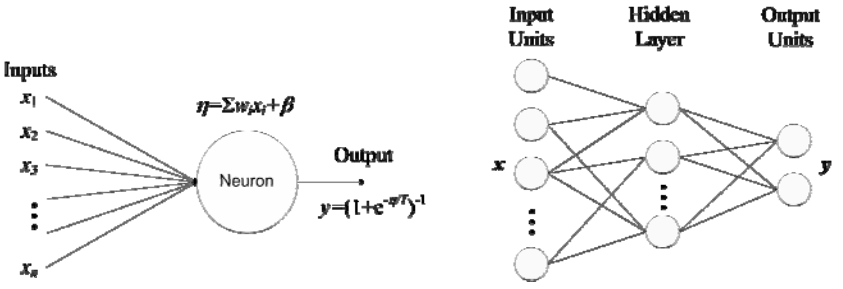
The basic structure in a neural network [39,40] is the neuron (or single-unit perceptron). A neuron performs an affine transformation followed by a nonlinear operation (see Fig. 3.5(a)). If the inputs to a neuron are denoted as $x_1, \ldots, x_n$, the neuron output y is computed as

$$y = \frac{1}{1 + \exp(-\eta / T)} \ , \tag{3.18}$$

where $\eta = w_1 x_1 + \ldots + w_n x_n + \gamma$, with $w_1, \ldots, w_n$ being regression coefficients. Here, γ is the bias value of a neuron, and T is a user-defined (slope) parameter. Neurons can be combined in multiple ways [39]. The most common neural network architecture is the multi-layer feed-forward network (see Fig. 3.5(b)).

The construction of a functional surrogate based on a neural network requires two main steps: (i) architecture selection, and (ii) network training. The network training can be stated as a nonlinear least-squares regression problem for a number of training points. Since the optimization cost function is nonlinear in all the optimization variables (neurons coefficients), the solution cannot be written using a closed-form expression, as it was the case before in (3.5) or in (3.9). A very popular technique for solving this regression problem is the error back-propagation algorithm [10,39]. If the network architecture is sufficiently complex, a neural network

can approximate a general set of functions [10]. However, in complicated cases (e.g., nonsmooth functions with a large number of variables) the underlying regression problem may be significantly involved.



(a) (b)

Fig. 3.5. Neural networks: (a) neuron basic structure; (b) two-layer feed-forward neural network architecture.

3.3.2.5 Other Techniques

The techniques described in this section refer to some other approaches that are gaining popularity recently. One of the most prominent approaches, which has been observed as a very general approximation tool, is support vector regression (SVR) [41,42]. SVR resorts to quadratic programming for a robust solving of the underlying optimization in the approximation procedure [43]. SVR is a variant of the support vector machines (SVMs) methodology developed by Vapnik [44], which was originally applied to classification problems. SVR/SVM exploits the structural risk minimization (SRM) principle, which has been shown (see, e.g., [41]) to be superior to the traditional empirical risk minimization (ERM) principle employed by several modeling technologies (e.g., neural networks). ERM is based on minimizing an error function for the set of training points. When the model structure is complex (e.g., higher order polynomials), ERM-based surrogates often result in overfitting. SRM incorporates the model complexity in the regression, and therefore yields surrogates that may be more accurate outside of the training set.

Moving least squares (MLS) [45] is a technique particularly popular in aerospace engineering. MLS is formulated as weighted least squares (WLS) [46]. In MLS, the error contribution from each training point $\boldsymbol{x}^{(i)}$ is multiplied by a weight ω_i that depends on the distance between $\boldsymbol{x}$ and $\boldsymbol{x}^{(i)}$. A common choice for the weights is

$$\omega_i(\| \boldsymbol{x}-\boldsymbol{x}^{(i)} \|) = \exp(-\| \boldsymbol{x}-\boldsymbol{x}^{(i)} \|^2) \ . \tag{3.19}$$

MLS is essentially an adapting surrogate, and this additional flexibility can be translated in more appealing designs (especially in computer graphics applications). However, MLS is computationally more expensive than WLS, since computing the approximation for each point $\boldsymbol{x}$ requires solving a new optimization problem.

Gaussian process regression (GPR) [47] is a surrogate modeling technique that, as kriging, addresses the approximation problem from a stochastic point view. From this perspective, and since Gaussian processes are mathematically tractable, it is relatively easy to compute error estimations for GPR-based surrogates in the form of uncertainty distributions. Under appropriate conditions, Gaussian processes can be shown to be equivalent to large neural networks [47]. Nevertheless, Gaussian process modeling typically requires much less regression parameters than neural networks.

3.3.3 Model Validation

Some of the methodologies described above determine a surrogate model together with some estimation of the attendant approximation error (e.g., kriging or Gaussian process regression). Alternatively, there are procedures that can be used in a stand-alone manner to validate the prediction capability of a given model beyond the set of training points. A simple way for validating a model is the split-sample method [3]. In this algorithm, the set of available data samples is divided into two subsets. The first subset is called the training subset and contains the points considered for the construction of the surrogate. The second subset is the testing subset and serves purely as a model validation objective. In general, the error estimated by a split-sample method depends strongly on how the set of data samples is partitioned. We also note that in this approach the samples available do not appear to be put to good use, since the surrogate is based on only a subset of them.

Cross-validation [3,48] is an extremely popular methodology for verifying the prediction capabilities of a model generated from a set of samples. In cross-validation the data set is divided into L subsets, and each of these subsets is sequentially used as testing set for a surrogate constructed on the other L–1 subsets. If the number of subsets L is equal to the sample size p, the approach is called leave-one-out cross-validation [3]. The prediction error can be estimated with all the L error measures obtained in this process (for example, as an average value). Cross-validation provides an error estimation that is less biased than with the split-sample method [3]. The disadvantage of this method is that the surrogate has to be constructed more than once. However, having multiple approximations may improve the robustness of the whole surrogate generation and validation approach, since all the data available is used with both training and testing purposes.

3.3.4 Surrogate Correction

In the first stages on any surrogate-based optimization procedure, it is desirable to use a surrogate that is valid globally in the search space [4] in order to avoid being trapped in local solutions with unacceptable cost function values. Once the search starts becoming local, the global accuracy of the initial surrogate may not be beneficial for making progress in the optimization[1]. For this reason, surrogate correction is crucial within any SBO methodology.

[1] As mentioned in Section 3.2, when solving the original optimization problem in (3.1) using a surrogate-based optimization framework, zero- and first-order local consistency conditions are essential for obtaining convergence to a first-order stationary point.

In this section we will describe two strategies for improving surrogates locally. The corrections described in Section 3.3.4.1 are based on mapping objective function values. In some occasions, the cost function can be expressed as a function of a model response. Section 3.3.4.2 presents the space-mapping concept that gives rise to a whole surrogate-based optimization paradigm (see Section 3.4.2).

3.3.4.1 Objective Function Correction

Most of the objective function corrections used in practice can be identified in one of these three groups: compositional, additive or multiplicative corrections. We will briefly illustrate each of these categories for correcting the surrogate $s^{(i)}(\boldsymbol{x})$, and discuss if zero- and first-order consistency conditions with $f(\boldsymbol{x})$ [7] can be satisfied.

The following compositional correction [20]

$$s^{(i+1)}(\boldsymbol{x}) = g(s^{(i)}(\boldsymbol{x})) \tag{3.20}$$

represents a simple scaling of the objective function. Since the mapping g is a real-valued function of a real variable, a compositional correction will not in general yield first-order consistency conditions. By selecting a mapping g that satisfies

$$g'(s^{(i)}(\boldsymbol{x}^{(i)})) = \frac{\nabla f(\boldsymbol{x}^{(i)}) \nabla s^{(i)}(\boldsymbol{x}^{(i)})^T}{\nabla s^{(i)}(\boldsymbol{x}^{(i)}) \nabla s^{(i)}(\boldsymbol{x}^{(i)})^T}, \tag{3.21}$$

the discrepancy between $\nabla f(\boldsymbol{x}^{(i)})$ and $\nabla s^{(i+1)}(\boldsymbol{x}^{(i)})$ (expressed in Euclidean norm) is minimized. It should be noticed that the correction in (3.21), as many transformations that ensure first-order consistency, requires a high-fidelity gradient, which may be expensive to compute. However, numerical estimates of $\nabla f(\boldsymbol{x}^{(i)})$ may yield in practice acceptable results.

The compositional correction can be also introduced in the parameter space [1]

$$s^{(i+1)}(\boldsymbol{x}) = s^{(i)}(\boldsymbol{p}(\boldsymbol{x})). \tag{3.22}$$

If $f(\boldsymbol{x}^{(i)})$ is not in the range of $s^{(i)}(\boldsymbol{x})$, then the condition $s^{(i)}(\boldsymbol{p}(\boldsymbol{x}^{(i)})) = f(\boldsymbol{x}^{(i)})$ is not achievable. We can overcome that issue by combining both compositional corrections. In that case, the following selection for g and $\boldsymbol{p}$

$$g(t) = t - s^{(i)}(\boldsymbol{x}^{(i)}) + f(\boldsymbol{x}^{(i)}), \tag{3.23}$$

$$\boldsymbol{p}(\boldsymbol{x}) = \boldsymbol{x}^{(i)} + \boldsymbol{J}_p(\boldsymbol{x} - \boldsymbol{x}^{(i)}), \tag{3.24}$$

where $\boldsymbol{J}_p$ is a $n \times n$ matrix for which $\boldsymbol{J}_p^T \nabla s^{(i)} = \nabla f(\boldsymbol{x}^{(i)})$, guarantees consistency.

Additive and multiplicative corrections allow obtaining first-order consistency conditions. For the additive case we can generally express the correction as

$$s^{(i+1)}(\boldsymbol{x}) = \lambda(\boldsymbol{x}) + s^{(i)}(\boldsymbol{x}). \tag{3.25}$$

The associated consistency conditions require that $\lambda(\boldsymbol{x})$ satisfies

$$\lambda(\boldsymbol{x}^{(i)}) = f(\boldsymbol{x}^{(i)}) - s^{(i)}(\boldsymbol{x}^{(i)}), \tag{3.26}$$

and

$$\nabla\lambda(\boldsymbol{x}^{(i)}) = \nabla f(\boldsymbol{x}^{(i)}) - \nabla s^{(i)}(\boldsymbol{x}^{(i)}). \tag{3.27}$$

Those requirements can be obtained by the following linear additive correction:

$$s^{(i+1)}(\boldsymbol{x}) = f(\boldsymbol{x}^{(i)}) - s^{(i)}(\boldsymbol{x}^{(i)}) + (\nabla f(\boldsymbol{x}^{(i)}) - \nabla s^{(i)}(\boldsymbol{x}^{(i)}))(\boldsymbol{x} - \boldsymbol{x}^{(i)}) + s^{(i)}(\boldsymbol{x}). \tag{3.28}$$

Multiplicative corrections (also known as the β-correlation method [20]) can be represented generically by

$$s^{(i+1)}(\boldsymbol{x}) = \alpha(\boldsymbol{x}) s^{(i)}(\boldsymbol{x}). \tag{3.29}$$

Assuming that $s^{(i)}(\boldsymbol{x}^{(i)}) \neq 0$, zero- and first-order consistency can be achieved if

$$\alpha(\boldsymbol{x}^{(i)}) = \frac{f(\boldsymbol{x}^{(i)})}{s^{(i)}(\boldsymbol{x}^{(i)})}, \tag{3.30}$$

and

$$\nabla\alpha(\boldsymbol{x}^{(i)}) = [\nabla f(\boldsymbol{x}^{(i)}) - f(\boldsymbol{x}^{(i)}) / s^{(i)}(\boldsymbol{x}^{(i)}) \nabla s^{(i)}(\boldsymbol{x}^{(i)})] / s^{(i)}(\boldsymbol{x}^{(i)}). \tag{3.31}$$

The requirement $s^{(i)}(\boldsymbol{x}^{(i)}) \neq 0$ is not strong in practice since very often the range of $f(\boldsymbol{x})$ (and thus, of the surrogate $s^{(i)}(\boldsymbol{x})$) is known beforehand, and hence, a bias can be introduced both for $f(\boldsymbol{x})$ and $s^{(i)}(\boldsymbol{x})$ to avoid cost function values equal to zero. In these circumstances the following multiplicative correction

$$s^{(i+1)}(\boldsymbol{x}) = \left[\frac{f(\boldsymbol{x}^{(i)})}{s^{(i)}(\boldsymbol{x}^{(i)})} + \frac{\nabla f(\boldsymbol{x}^{(i)}) s^{(i)}(\boldsymbol{x}^{(i)}) - f(\boldsymbol{x}^{(i)}) \nabla s^{(i)}(\boldsymbol{x}^{(i)})}{(s^{(i)}(\boldsymbol{x}^{(i)}))^2} (\boldsymbol{x} - \boldsymbol{x}^{(i)}) \right] s^{(i)}(\boldsymbol{x}), \tag{3.32}$$

is consistent with conditions (3.30) and (3.31).

3.3.4.2 Space Mapping Concept

Space mapping (SM) [1,5,6] is a well-known methodology for correcting a given (either functional or physical) surrogate. SM algorithms aim at objective functions $f(\boldsymbol{x})$ that can be written as a functional U of a so-called system response $\boldsymbol{R}_f(\boldsymbol{x}) \in R^m$

$$f(\boldsymbol{x}) = U(\boldsymbol{R}_f(\boldsymbol{x})). \tag{3.33}$$

The fine model response $\boldsymbol{R}_f(\boldsymbol{x})$ is assumed to be accurate but computationally expensive. The coarse model response $\boldsymbol{R}_c(\boldsymbol{x}) \in R^m$ is much cheaper to evaluate than the fine model response at the expense of being an approximation of it. SM establishes a correction between model responses rather than between objective functions. The corrected model response will be denoted as $\boldsymbol{R}_s(\boldsymbol{x}; \boldsymbol{p}_{SM}) \in R^m$, and $\boldsymbol{p}_{SM}$ represents a set of parameters that describes the type of correction performed.

We can find in the literature four different groups of coarse model response corrections [1,5]:

1. Input space mapping [1]. The response correction is based on an affine transformation on the low-fidelity model parameter space. Example: $\boldsymbol{R}_s(\boldsymbol{x}; \boldsymbol{p}_{SM}) = \boldsymbol{R}_s(\boldsymbol{x}; \boldsymbol{B},\boldsymbol{c}) = \boldsymbol{R}_c(\boldsymbol{B}\,\boldsymbol{x} + \boldsymbol{c})$.
2. Output space mapping [5]. The response correction is based on an affine transformation on the low-fidelity model response. Example: $\boldsymbol{R}_s(\boldsymbol{x}; \boldsymbol{p}_{SM}) = \boldsymbol{R}_s(\boldsymbol{x}; \boldsymbol{A},\boldsymbol{d}) = \boldsymbol{A}\,\boldsymbol{R}_c(\boldsymbol{x})+\boldsymbol{d}$. Manifold-mapping (see Section 3.4.3) is a particular case of output space mapping.
3. Implicit space mapping [49]. In some cases, there are additional parameters $\boldsymbol{x}_p \in R^{n_p}$ in the coarse model response $\boldsymbol{R}_c(\boldsymbol{x}; \boldsymbol{x}_p)$ that can be tuned for better aligning of the fine and coarse model responses. Example: $\boldsymbol{R}_s(\boldsymbol{x}; \boldsymbol{p}_{SM}) = \boldsymbol{R}_s(\boldsymbol{x}; \boldsymbol{x}_p) = \boldsymbol{R}_c(\boldsymbol{x}; \boldsymbol{x}_p)$. These additional parameters are known in SM lexicon as pre-assigned parameters, and are in general different from the optimization variables $\boldsymbol{x}$.
4. Custom corrections that exploit the structure of the given design problem [1]. In many occasions the model responses are obtained through the sweeping of some parameter t:

$$\boldsymbol{R}_f(\boldsymbol{x}) = \boldsymbol{R}_f(\boldsymbol{x};t) = [R_f(\boldsymbol{x};t_1) \quad R_f(\boldsymbol{x};t_2) \quad \ldots \quad R_f(\boldsymbol{x};t_m)]^T, \tag{3.34}$$

$$\boldsymbol{R}_c(\boldsymbol{x}) = \boldsymbol{R}_c(\boldsymbol{x};t) = [R_c(\boldsymbol{x};t_1) \quad R_c(\boldsymbol{x};t_2) \quad \ldots \quad R_c(\boldsymbol{x};t_m)]^T. \tag{3.35}$$

   Examples of this situation appear when the parameter t represents time or frequency. The response correction considered in this case[2] could be based on an affine transformation on the sweeping parameter space:

$$\boldsymbol{R}_s(\boldsymbol{x}; \boldsymbol{p}_{SM}) = \boldsymbol{R}_s(\boldsymbol{x}; r_0, r_1) = \boldsymbol{R}_c(\boldsymbol{x}; r_0 + r_1 t). \tag{3.36}$$

In Fig. 3.6 we illustrate by means of block diagrams the four SM-based correction strategies introduced above, together with a combination of three of them.

The surrogate response is usually optimized with respect to the SM parameters $\boldsymbol{p}_{SM}$ in order to reduce the model discrepancy for all or part of the data available $\boldsymbol{R}_f(\boldsymbol{x}^{(1)})$, $\boldsymbol{R}_f(\boldsymbol{x}^{(2)})$, … , $\boldsymbol{R}_f(\boldsymbol{x}^{(p)})$:

$$\boldsymbol{p}_{SM} = \arg\min_{\boldsymbol{p}_{SM}} \sum_{k=1}^{p} \omega^{(k)} \| \boldsymbol{R}_f(\boldsymbol{x}^{(k)}) - \boldsymbol{R}_s(\boldsymbol{x}^{(k)}; \boldsymbol{p}_{SM}) \|, \tag{3.37}$$

where $0 \le \omega^{(k)} \le 1$ are weights for each of the samples. The corrected surrogate $\boldsymbol{R}_s(\boldsymbol{x}; \boldsymbol{p}_{SM})$ can be used as an approximation to the fine response $\boldsymbol{R}_f(\boldsymbol{x})$ in the vicinity of the sampled data. The minimization in (3.37) is known in SM literature as parameter extraction [1]. The solving of this optimization process is not exempt from difficulties, since in many cases the problem is ill-conditioned. We can find in [1] a number of techniques for addressing parameter extraction in a robust manner.

[2] This type of space mapping is known as frequency space mapping [4], and it was originally proposed in microwave engineering applications (in these applications t usually refers to frequency).

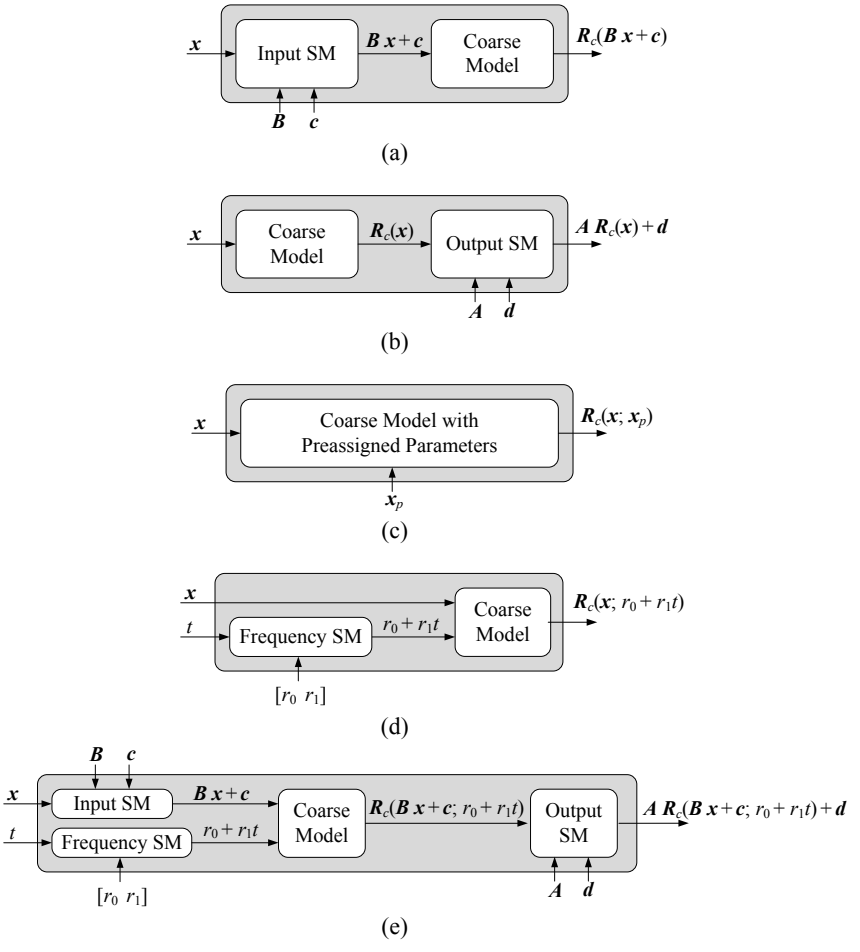


Fig. 3.6 Basic space-mapping surrogate correction types: (a) input SM, (b) output SM, (c) implicit SM, (d) frequency SM, and (e) composite using input, output and frequency SM.

3.4 Surrogate-Based Optimization Techniques

In this section, we will introduce several optimization strategies that exploit surrogate models. More specifically, we will describe approximation model management optimization [7], space mapping [5], manifold mapping [8], and the surrogate management framework [9]. The first three approaches follow the surrogate-based optimization framework presented in Section 3.2. We will conclude the section with a brief discussion on addressing the tradeoff between exploration and/or exploitation in the optimization process.

3.4.1 Approximation Model Management Optimization

Approximation model management optimization (AMMO) [7] relies on trust-region gradient-based optimization combined with the multiplicative linear surrogate correction (3.32) introduced in Section 3.3.4.1.

The basic AMMO algorithm can be summarized as follows:

1. Set initial guess $\boldsymbol{x}^{(0)}$, $s^{(0)}(\boldsymbol{x})$, and $i = 0$, and select the initial trust-region radius $\delta > 0$.
2. If $i > 0$, then $s^{(i)}(\boldsymbol{x}) = \alpha(\boldsymbol{x})\, s^{(i-1)}(\boldsymbol{x})$.
3. Solve $\boldsymbol{h}^* = \text{argmin}\, s^{(i)}(\boldsymbol{x}^{(i)} + \boldsymbol{h})$ subject to $\|\boldsymbol{h}\|_\infty \leq \delta$.
4. Calculate $\rho = (f(\boldsymbol{x}^{(i)}) - f(\boldsymbol{x}^{(i)} + \boldsymbol{h}^*))/(s(\boldsymbol{x}^{(i)}) - s^{(i)}(\boldsymbol{x}^{(i)} + \boldsymbol{h}^*))$.
5. If $f(\boldsymbol{x}^{(i)}) > f(\boldsymbol{x}^{(i)} + \boldsymbol{h}^*)$, then set $\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + \boldsymbol{h}^*$; otherwise $\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)}$.
6. Update the search radius δ based on the value of ρ.
7. Set $i = i + 1$, and if the termination condition is not satisfied, go to Step 2.

Additional constraints can also be incorporated in the optimization through Step 3. AMMO can also be extended to cases where the constraints are expensive to evaluate and can be approximated by surrogates [50]. The search radius δ is updated using the standard trust-region rules [11,51]. We reiterate that the surrogate correction considered yields zero- and first-order consistency with $f(\boldsymbol{x})$. Since this surrogate-based approach is safeguarded by means of a trust-region method, the whole scheme can be proven to be globally convergent to a first-order stationary point of the original optimization problem (3.1).

3.4.2 Space Mapping

The space mapping (SM) paradigm [1,5] was originally developed in microwave engineering optimal design applications, and gave rise to an entire family of surrogate-based optimization approaches. Nowadays, its popularity is spreading across several engineering disciplines [52,53,1]. The initial space-mapping optimization methodologies were based on input SM [1], i.e., a linear correction of the coarse model design space. This kind of correction is well suited for many engineering problems, particularly in electrical engineering, where the model discrepancy is mostly due to second-order effects (e.g., presence of parasitic components). In these applications the model response ranges are often similar in shape, but slightly distorted and/or shifted with respect to a sweeping parameter (e.g., signal frequency).

Space mapping can be incorporated in the SBO framework by just identifying the sequence of surrogates with

$$s^{(0)}(\boldsymbol{x}) = U(R_c(\boldsymbol{x}))\,, \tag{3.38}$$

and

$$s^{(i)}(\boldsymbol{x}) = U(R_s(\boldsymbol{x}; \boldsymbol{p}_{SM}^{(i)}))\,, \tag{3.39}$$

for $i > 0$. The parameters $\boldsymbol{p}_{SM}^{(i)}$ are obtained by parameter extraction as in (3.37). The accuracy of the corrected surrogate will clearly depend on the quality of the coarse model response [16]. In microwave design applications it has been many times observed that the number of points p needed for obtaining a satisfactory SM-based corrected surrogate is on the order of the number of optimization variables n [1]. Though output SM can be used to obtain both zero- and first-order consistency conditions with $f(\boldsymbol{x})$, many other SM-based optimization algorithms that have been applied in practice do not satisfy those conditions, and in some occasions convergence problems have been identified [14]. Additionally, the choice of an adequate SM correction approach is not always obvious [14]. However, in multiple occasions and in several different disciplines [52,53,1], space mapping has been reported as a very efficient means for obtaining satisfactory optimal designs.

Convergence properties of space-mapping optimization algorithms can be improved when these are safeguarded by a trust region [54]. Similarly to AMMO, the SM surrogate model optimization is restricted to a neighborhood of $\boldsymbol{x}^{(i)}$ (this time by using the Euclidean norm) as follows

$$\boldsymbol{x}^{(i+1)} = \arg\min_{\boldsymbol{x}} s^{(i)}(\boldsymbol{x}) \quad \text{subject to} \quad \| \boldsymbol{x} - \boldsymbol{x}^{(i)} \|_2 \leq \delta^{(i)}, \tag{3.40}$$

where $\delta^{(i)}$ denotes the trust-region radius at iteration i. The trust region is updated at every iteration by means of precise criteria [11]. A number of enhancements for space mapping have been suggested recently in the literature (e.g., zero-order and aproximate/exact first order consistency conditions with $f(\boldsymbol{x})$ [54], or adaptively constrained parameter extraction [55]).

The quality of a surrogate within space mapping can be assessed by means of the techniques described in [14,16]. These methods are based on evaluating the high-fidelity model at several points (and thus, they require some extra computational effort). With that information, some conditions required for convergence are approximated numerically, and as a result, low-fidelity models can be compared based on these approximate conditions. The quality assessment algorithms presented in [14,16] can also be embedded into SM optimization algorithms in order to throw some light on the delicate issue of selecting the most adequate SM surrogate correction.

It should be emphasized that space mapping is not a general-purpose optimization approach. The existence of the computationally cheap and sufficiently accurate low-fidelity model is an important prerequisite for this technique. If such a coarse model does exist, satisfactory designs are often obtained by space mapping after a relatively small number of evaluations of the high-fidelity model. This number is usually on the order of the number of optimization variables n [14], and very frequently represents a dramatic reduction in the computational cost required for solving the same optimization problem with other methods that do not rely on surrogates. In the absence of the above-mentioned low-fidelity model, space-mapping optimization algorithms may not perform efficiently.

3.4.3 Manifold Mapping

Manifold mapping (MM) [8,56] is a particular case of output space mapping, that is supported by convergence theory [13,56], and does not require the parameter extraction step shown in (3.37). Manifold mapping can be integrated in the SBO framework by just considering $s^{(i)}(\boldsymbol{x}) = U(\boldsymbol{R}_s^{(i)}(\boldsymbol{x}))$ with the response correction for $i \geq 0$ defined as

$$\boldsymbol{R}_s^{(i)}(\boldsymbol{x}) = \boldsymbol{R}_f(\boldsymbol{x}^{(i)}) + \boldsymbol{S}^{(i)}\left(\boldsymbol{R}_c(\boldsymbol{x}) - \boldsymbol{R}_c(\boldsymbol{x}^{(i)})\right), \tag{3.41}$$

where $\boldsymbol{S}^{(i)}$, for $i \geq 1$, is the following $m \times m$ matrix

$$\boldsymbol{S}^{(i)} = \Delta\boldsymbol{F}\Delta\boldsymbol{C}^{\dagger}, \tag{3.42}$$

with

$$\Delta\boldsymbol{F} = [\boldsymbol{R}_f(\boldsymbol{x}^{(i)}) - \boldsymbol{R}_f(\boldsymbol{x}^{(i-1)}) \quad \ldots \quad \boldsymbol{R}_f(\boldsymbol{x}^{(i)}) - \boldsymbol{R}_f(\boldsymbol{x}^{(\max\{i-n,0\})})], \tag{3.43}$$

$$\Delta\boldsymbol{C} = [\boldsymbol{R}_c(\boldsymbol{x}^{(i)}) - \boldsymbol{R}_c(\boldsymbol{x}^{(i-1)}) \quad \ldots \quad \boldsymbol{R}_c(\boldsymbol{x}^{(i)}) - \boldsymbol{R}_c(\boldsymbol{x}^{(\max\{i-n,0\})})]. \tag{3.44}$$

The matrix $\boldsymbol{S}^{(0)}$ is typically taken as the identity matrix $\boldsymbol{I}_m$. Here, † denotes the pseudoinverse operator defined for $\Delta\boldsymbol{C}$ as

$$\Delta\boldsymbol{C}^{\dagger} = \boldsymbol{V}_{\Delta\boldsymbol{C}}\boldsymbol{\Sigma}_{\Delta\boldsymbol{C}}^{\dagger}\boldsymbol{U}_{\Delta\boldsymbol{C}}^{T}, \tag{3.45}$$

where $\boldsymbol{U}_{\Delta\boldsymbol{C}}$, $\boldsymbol{\Sigma}_{\Delta\boldsymbol{C}}$, and $\boldsymbol{V}_{\Delta\boldsymbol{C}}$ are the factors in the singular value decomposition of $\Delta\boldsymbol{C}$. The matrix $\boldsymbol{\Sigma}_{\Delta\boldsymbol{C}}{}^{\dagger}$ is the result of inverting the nonzero entries in $\boldsymbol{\Sigma}_{\Delta\boldsymbol{C}}$, leaving the zeroes invariant [8]. Some mild general assumptions on the model responses are made in theory [56] so that every pseudoinverse introduced is well defined.

The response correction $\boldsymbol{R}_s^{(i)}(\boldsymbol{x})$ is an approximation of

$$\boldsymbol{R}_s^{*}(\boldsymbol{x}) = \boldsymbol{R}_f(\boldsymbol{x}^{*}) + \boldsymbol{S}^{*}(\boldsymbol{R}_c(\boldsymbol{x}) - \boldsymbol{R}_c(\boldsymbol{x}^{*})), \tag{3.46}$$

with $\boldsymbol{S}^{*}$ being the $m \times m$ matrix defined as

$$\boldsymbol{S}^{*} = J_f(\boldsymbol{x}^{*})J_c^{\dagger}(\boldsymbol{x}^{*}), \tag{3.47}$$

where $J_f(\boldsymbol{x}^{*})$ and $J_c(\boldsymbol{x}^{*})$ stand for the fine and coarse model response Jacobian, respectively, evaluated at $\boldsymbol{x}^{*}$. Obviously, neither $\boldsymbol{x}^{*}$ nor $\boldsymbol{S}^{*}$ is known beforehand. Therefore, one needs to use an iterative approximation, such as the one in (3.41)-(3.45), in the actual manifold-mapping algorithm.

The manifold-mapping model alignment is illustrated in Fig. 3.7 for the least-squares optimization problem

$$U(\boldsymbol{R}_f(\boldsymbol{x})) = \| \boldsymbol{R}_f(\boldsymbol{x}) - \boldsymbol{y} \|_2^2, \tag{3.48}$$

with $\boldsymbol{y} \in R^m$ being the design specifications given. In that figure the point $\boldsymbol{x}_c^{*}$ denotes the minimizer corresponding to the coarse model cost function $U(\boldsymbol{R}_c(\boldsymbol{x}))$. We note that, in absence of constraints, the optimality associated to (3.48) is translated into the orthogonality between the tangent plane for $\boldsymbol{R}_f(\boldsymbol{x})$ at $\boldsymbol{x}^{*}$ and the vector $\boldsymbol{R}_f(\boldsymbol{x}^{*}) - \boldsymbol{y}$.

If the low-fidelity model has a negligible computational cost when compared to the high-fidelity one, the MM surrogate can be explored globally. The MM algorithm is in this case endowed with some robustness with respect to being trapped in unsatisfactory local minima.

For least-squares optimization problems as in (3.48), manifold mapping is supported by mathematically sound convergence theory [13]. We can identify four factors relevant for the convergence of the scheme above to the fine model optimizer $\boldsymbol{x}^*$:

1. The model responses being smooth.
2. The surrogate optimization in (3.2) being well-posed.
3. The discrepancy of the optimal model response $\boldsymbol{R}_f(\boldsymbol{x}^*)$ with respect to the design specifications being sufficiently small.
4. The low-fidelity model response being a sufficiently good approximation of the high-fidelity model response.

In most practical situations the requirements associated to the first three factors are satisfied, and since the low-fidelity models often considered are based on expert knowledge accumulated over the years, the similarity between the model responses can be frequently good enough for having convergence.

Manifold-mapping algorithms can be expected to converge for a merit function U sufficiently smooth. Since the correction in (3.41) does not involve U, if the model responses are smooth enough, and even when U is not differentiable, manifold mapping may still yield satisfactory solutions. The experimental evidence given in [57] for designs based on minimax objective functions indicates that the MM approach can be used successfully in more general situations than those for which theoretical results have been obtained.

The basic manifold-mapping algorithm can be modified in a number of ways. Convergence appears to improve if derivative information is introduced in the algorithm [13]. The incorporation of a Levenberg-Marquardt strategy in manifold mapping [58] can be seen as a convergence safeguard analogous to a trust-region method [11]. Manifold mapping can also be extended to designs where the constraints are determined by time-consuming functions, and for which surrogates are available as well [59].

3.4.4 Surrogate Management Framework

The surrogate management framework (SMF) [9] is mainly based on pattern search. Pattern search [60] is a general set of derivative-free optimizers that can be proven to be globally convergent to first-order stationary points. A pattern search optimization algorithm is based on exploring the search space by means of a structured set of points (pattern or stencil) that is modified along iterations. The pattern search scheme considered in [9] has two main steps per iteration: search and poll. Each iteration starts with a pattern of size Δ centered at $\boldsymbol{x}^{(i)}$. The search step is optional and is always performed before the poll step. In the search stage a (small) number of points are selected from the search space (typically by means of a surrogate), and the cost function $f(\boldsymbol{x})$ is evaluated at these points. If the cost function

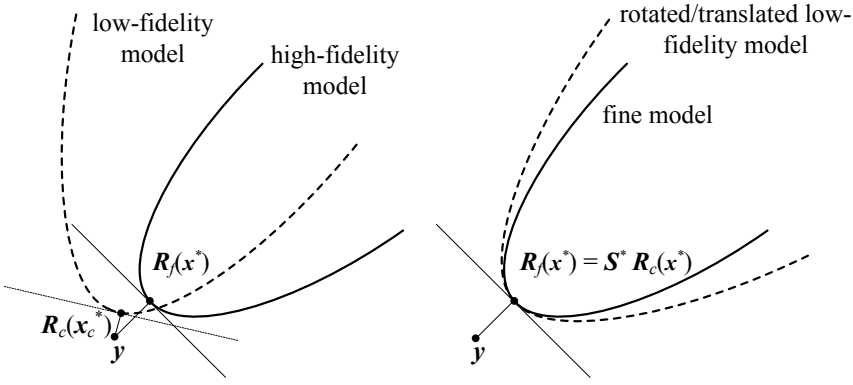


Fig. 3.7 Illustration of the manifold-mapping model alignment for a least-squares optimization problem. The point $\boldsymbol{x}_c^*$ denotes the minimizer corresponding to the coarse model response, and the point $\boldsymbol{y}$ is the vector of design specifications. Thin solid and dashed straight lines denote the tangent planes for the fine and coarse model response at their optimal designs, respectively. By the linear correction $\boldsymbol{S}^*$, the point $\boldsymbol{R}_c(\boldsymbol{x}^*)$ is mapped to $\boldsymbol{R}_f(\boldsymbol{x}^*)$, and the tangent plane for $\boldsymbol{R}_c(\boldsymbol{x})$ at $\boldsymbol{R}_c(\boldsymbol{x}^*)$ to the tangent plane for $\boldsymbol{R}_f(\boldsymbol{x})$ at $\boldsymbol{R}_f(\boldsymbol{x}^*)$ [13].

for some of them improves on $f(\boldsymbol{x}^{(i)})$ the search step is declared successful, the current pattern is centered at this new point, and a new search step is started. Otherwise a poll step is taken. Polling requires computing $f(\boldsymbol{x})$ for points in the pattern. If one of these points is found to improve on $f(\boldsymbol{x}^{(i)})$, the poll step is declared successful, the pattern is translated to this new point, and a new search step is performed. Otherwise the whole pattern search iteration is considered unsuccessful and the termination condition is checked. This stopping criterion is typically based on the pattern size Δ [9,61]. If, after the unsuccessful pattern search iteration another iteration is needed, the pattern size Δ is decreased, and a new search step is taken with the pattern centered again at $\boldsymbol{x}^{(i)}$. Surrogates are incorporated in the SMF through the search step. For example, kriging (with Latin hypercube sampling) is considered in the SMF application studied in [61].

In order to guarantee convergence to a stationary point, the set of vectors formed by each pattern point and the pattern center should be a generating (or positive spanning) set [60,61]. A generating set for R^n consists of a set of vectors whose non-negative linear combinations span R^n. Generating sets are crucial in proving convergence (for smooth objective functions) due to the following property: if a generating set is centered at $\boldsymbol{x}^{(i)}$ and $\nabla f(\boldsymbol{x}^{(i)}) \neq 0$, then at least one of the vectors in the generating set defines a descent direction [60]. Therefore, if $f(\boldsymbol{x})$ is smooth and $\nabla f(\boldsymbol{x}^{(i)}) \neq 0$, we can expect that for a pattern size Δ small enough, some of the points in the associated stencil will improve on $f(\boldsymbol{x}^{(i)})$.

Though pattern search optimization algorithms typically require many more function evaluations than gradient-based techniques, the computations in both the search and poll steps can be performed in a distributed fashion. On top of that, the use of surrogates, as is the case for the SMF, generally accelerates noticeably the entire optimization process.

3.4.5 Exploitation versus Exploration

The surrogate-based optimization framework starts from an initial surrogate model which is updated using the high-fidelity model data that is accumulated in the optimization process. In particular, the high-fidelity model has to be evaluated for verification at any new design $\boldsymbol{x}^{(i)}$ provided by the surrogate model. The new points at which we evaluate the high-fidelity model are sometimes referred to as infill points [4]. We reiterate that this data can be used to enhance the surrogate. The selection of the infill points is also known as adaptive sampling [4].

Infill points in approximation model management optimization, space mapping and manifold mapping are in practice selected through local optimization of the surrogate (global optimization for problems with a medium/large number of variables and even relatively inexpensive surrogates can be a time-consuming procedure). The new infill points in the surrogate management framework are taken based only on high-fidelity cost function improvement. As we have seen in this section, the four surrogate-based optimization approaches discussed are supported by local optimality theoretical results. In other words, these methodologies intrinsically aim at the exploitation of certain region of the design space (the neighborhood of a first-order stationary point). If the surrogate is valid globally, the first iterations of these four optimization approaches can be used to avoid being trapped in unsatisfactory local solutions (i.e., global exploration steps).

The exploration of the design space implies in most cases a global search. If the underlying objective function is non-convex, exploration usually boils down to performing a global sampling of the search space, for example, by selecting those points that maximize some estimation of the error associated to the surrogate considered [4]. It should be stressed that global exploration is often impractical, especially for computationally expensive cost functions with a medium/large number of optimization variables (more than a few tens). Additionally, pure exploration may not be a good approach for updating the surrogate in an optimization context, since a great amount of computing resources can be spent in modeling parts of the search space that are not interesting from an optimal design point of view.

Therefore, it appears that in optimization there should be a balance between exploitation and exploration. As suggested in [4], this tradeoff could be formulated in the context of surrogate-based optimization, for example, by means of a bi-objective optimization problem (with global measure of the error associated to the surrogate as second objective function), by maximizing the probability of improvement upon the best observed objective function value, or through the maximization of the expected cost function improvement. As mentioned above, these hybrid approaches will find difficulties in performing an effective global search in designs with a medium/large number of optimization variables.

3.5 Final Remarks

In this chapter, an overview of surrogate modeling, with an emphasis on optimization, has been presented. Surrogate-based optimization plays an important role in

contemporary engineering design, and the importance of this role will most likely increase in the near future. One of the reasons for this increase is the fact that computer simulations have become a major design tool in most engineering areas. In order for these simulations to be sufficiently accurate, more and more phenomena have to be captured. This level of sophistication renders simulations computationally expensive, particularly when they deal with the time-varying three-dimensional structures considered in many engineering fields. Hence, evaluation times of several days, or even weeks, are nowadays not uncommon. The direct use of CPU-intensive numerical models in some off-the-shelf automated optimization procedures (e.g., gradient-based techniques with approximate derivatives) is very often prohibitive. Surrogate-based optimization can be a very useful approach in this context, since, apart from reducing significantly the number of high-fidelity expensive simulations in the whole design process, it also helps in addressing important high-fidelity cost function issues (e.g., presence of discontinuities and/or multiple local optima).

References

1. Bandler, J.W., Cheng, Q.S., Dakroury, S.A., Mohamed, A.S., Bakr, M.H., Madsen, K., Søndergaard, J.: Space mapping: the state of the art. IEEE Trans. Microwave Theory Tech. 52, 337–361 (2004)
2. Pironneau, O.: On optimum design in fluid mechanics. J. Fluid Mech. 64, 97–110 (1974)
3. Queipo, N.V., Haftka, R.T., Shyy, W., Goel, T., Vaidynathan, R., Tucker, P.K.: Surrogate-based analysis and optimization. Progress in Aerospace Sciences 41, 1–28 (2005)
4. Forrester, A.I.J., Keane, A.J.: Recent advances in surrogate-based optimization. Prog. Aerospace Sciences 45, 50–79 (2009)
5. Koziel, S., Bandler, J.W., Madsen, K.: A space mapping framework for engineering optimization: theory and implementation. IEEE Trans. Microwave Theory Tech. 54, 3721–3730 (2006)
6. Koziel, S., Cheng, Q.S., Bandler, J.W.: Space mapping. IEEE Microwave Magazine 9, 105–122 (2008)
7. Alexandrov, N.M., Lewis, R.M.: An overview of first-order model management for engineering optimization. Optimization and Engineering 2, 413–430 (2001)
8. Echeverria, D., Hemker, P.W.: Space mapping and defect correction. CMAM Int. Mathematical Journal Computational Methods in Applied Mathematics 5, 107–136 (2005)
9. Booker, A.J., Dennis, J.E., Frank, P.D., Serafini, D.B., Torczon, V., Trosset, M.W.: A rigorous framework for optimization of expensive functions by surrogates. Structural Optimization 17, 1–13 (1999)
10. Simpson, T.W., Peplinski, J., Koch, P.N., Allen, J.K.: Metamodels for computer-based engineering design: survey and recommendations. Engineering with Computers 17, 129–150 (2001)
11. Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust Region Methods. MPS-SIAM Series on Optimization (2000)

12. Alexandrov, N.M., Dennis, J.E., Lewis, R.M., Torczon, V.: A trust region framework for managing use of approximation models in optimization. Struct. Multidisciplinary Optim. 15, 16–23 (1998)
13. Echeverría, D., Hemker, P.W.: Manifold mapping: a two-level optimization technique. Computing and Visualization in Science 11, 193–206 (2008)
14. Koziel, S., Bandler, J.W., Madsen, K.: Quality assessment of coarse models and surrogates for space mapping optimization. Optimization Eng. 9, 375–391 (2008)
15. Koziel, S., Bandler, J.W.: Coarse and surrogate model assessment for engineering design optimization with space mapping. In: IEEE MTT-S Int. Microwave Symp. Dig, Honolulu, HI, pp. 107–110 (2007)
16. Koziel, S., Bandler, J.W.: Space-mapping optimization with adaptive surrogate model. IEEE Trans. Microwave Theory Tech. 55, 541–547 (2007)
17. Alexandrov, N.M., Nielsen, E.J., Lewis, R.M., Anderson, W.K.: First-order model management with variable-fidelity physics applied to multi-element airfoil optimization. In: AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Design and Optimization, Long Beach, CA, AIAA Paper 2000-4886 (2000)
18. Wu, K.-L., Zhao, Y.-J., Wang, J., Cheng, M.K.K.: An effective dynamic coarse model for optimization design of LTCC RF circuits with aggressive space mapping. IEEE Trans. Microwave Theory Tech. 52, 393–402 (2004)
19. Robinson, T.D., Eldred, M.S., Willcox, K.E., Haimes, R.: Surrogate-based optimization using multifidelity models with variable parameterization and corrected space mapping. AIAA Journal 46, 2814–2822 (2008)
20. Søndergaard, J.: Optimization using surrogate models – by the space mapping technique. Ph.D. Thesis, Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby (2003)
21. Kleijnen, J.P.C.: Kriging metamodeling in simulation: a review. European Journal of Operational Research 192, 707–716 (2009)
22. Rayas-Sanchez, J.E.: EM-based optimization of microwave circuits using artificial neural networks: the state-of-the-art. IEEE Trans. Microwave Theory Tech. 52, 420–435 (2004)
23. Giunta, A.A., Wojtkiewicz, S.F., Eldred, M.S.: Overview of modern design of experiments methods for computational simulations. American Institute of Aeronautics and Astronautics, paper AIAA 2003–0649 (2003)
24. Santner, T.J., Williams, B., Notz, W.: The Design and Analysis of Computer Experiments. Springer, Heidelberg (2003)
25. Koehler, J.R., Owen, A.B.: Computer experiments. In: Ghosh, S., Rao, C.R. (eds.) Handbook of Statistics, vol. 13, pp. 261–308. Elsevier Science B.V., Amsterdam (1996)
26. Cheng, Q.S., Koziel, S., Bandler, J.W.: Simplified space mapping approach to enhancement of microwave device models. Int. J. RF and Microwave Computer-Aided Eng. 16, 518–535 (2006)
27. McKay, M., Conover, W., Beckman, R.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21, 239–245 (1979)
28. Beachkofski, B., Grandhi, R.: Improved distributed hypercube sampling. American Institute of Aeronautics and Astronautics, Paper AIAA 2002–1274 (2002)
29. Leary, S., Bhaskar, A., Keane, A.: Optimal orthogonal-array-based latin hypercubes. Journal of Applied Statistics 30, 585–598 (2003)

30. Ye, K.Q.: Orthogonal column latin hypercubes and their application in computer experiments. Journal of the American Statistical Association 93, 1430–1439 (1998)
31. Palmer, K., Tsui, K.-L.: A minimum bias latin hypercube design. IIE Transactions 33, 793–808 (2001)
32. Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd edn. The Johns Hopkins University Press, Baltimore (1996)
33. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-Free Optimization. MPS-SIAM Series on Optimization, MPS-SIAM (2009)
34. Wild, S.M., Regis, R.G., Shoemaker, C.A.: ORBIT: Optimization by radial basis function interpolation in trust-regions. SIAM J. Sci. Comput. 30, 3197–3219 (2008)
35. Journel, A.G., Huijbregts, C.J.: Mining Geostatistics. Academic Press, London (1981)
36. O'Hagan, A.: Curve fitting and optimal design for predictions. Journal of the Royal Statistical Society B 40, 1–42 (1978)
37. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press, Cambridge (2006)
38. Jones, D., Schonlau, M., Welch, W.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization 13, 455–492 (1998)
39. Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Prentice-Hall, Englewood Cliffs (1998)
40. Minsky, M.I., Papert, S.A.: Perceptrons: An Introduction to Computational Geometry. MIT Press, Cambridge (1969)
41. Gunn, S.R.: Support vector machines for classification and regression. Technical Report. School of Electronics and Computer Science, University of Southampton (1998)
42. Angiulli, G., Cacciola, M., Versaci, M.: Microwave devices and antennas modeling by support vector regression machines. IEEE Trans. Magn. 43, 1589–1592 (2007)
43. Smola, A.J., Schölkopf, B.: A tutorial on support vector regression. Statistics and Computing 14, 199–222 (2004)
44. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1995)
45. Levin, D.: The approximation power of moving least-squares. Mathematics of Computation 67, 1517–1531 (1998)
46. Aitken, A.C.: On least squares and linear combinations of observations. Proceedings of the Royal Society of Edinburgh 55, 42–48 (1935)
47. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press, Massachussets (2006)
48. Geisser, S.: Predictive Inference. Chapman and Hall, Boca Raton (1993)
49. Koziel, S., Cheng, Q.S., Bandler, J.W.: Implicit space mapping with adaptive selection of preassigned parameters. IET Microwaves, Antennas & Propagation 4, 361–373 (2010)
50. Alexandrov, N.M., Lewis, R.M., Gumbert, C.R., Green, L.L., Newman, P.A.: Approximation and model management in aerodynamic optimization with variable-fidelity models. AIAA Journal of Aircraft 38, 1093–1101 (2001)
51. Moré, J.J.: Recent developments in algorithms and software for trust region methods. In: Bachem, A., Grötschel, M., Korte, B. (eds.) Mathematical Programming. The State of Art, pp. 258–287. Springer, Heidelberg (1983)
52. Leary, S.J., Bhaskar, A., Keane, A.J.: A constraint mapping approach to the structural optimization of an expensive model using surrogates. Optimization and Engineering 2, 385–398 (2001)

53. Redhe, M., Nilsson, L.: Optimization of the new Saab 9-3 exposed to impact load using a space mapping technique. Structural and Multidisciplinary Optimization 27, 411–420 (2004)
54. Koziel, S., Bandler, J.W., Cheng, Q.S.: Robust trust-region space-mapping algorithms for microwave design optimization. IEEE Trans. Microwave Theory and Tech. 58, 2166–2174 (2010)
55. Koziel, S., Bandler, J.W., Cheng, Q.S.: Adaptively constrained parameter extraction for robust space mapping optimization of microwave circuits. IEEE MTT-S Int. Microwave Symp. Dig., 205–208 (2010)
56. Echeverría, D.: Multi-Level optimization: space mapping and manifold mapping. Ph.D. Thesis, Faculty of Science, University of Amsterdam (2007)
57. Koziel, S., Echeverría Ciaurri, D.: Reliable simulation-driven design optimization of microwave structures using manifold mapping. Progress in Electromagnetics Research B 26, 361–382 (2010)
58. Hemker, P.W., Echeverría, D.: A trust-region strategy for manifold mapping optimization. JCP Journal of Computational Physics 224, 464–475 (2007)
59. Echeverría, D.: Two new variants of the manifold-mapping technique. COMPEL The International Journal for Computation and Mathematics in Electrical Engineering 26, 334–344 (2007)
60. Kolda, T.G., Lewis, R.M., Torczon, V.: Optimization by direct search: new perspectives on some classical and modern methods. SIAM Review 45, 385–482 (2003)
61. Marsden, A.L., Wang, M., Dennis, J.E., Moin, P.: Optimal aeroacoustic shape design using the surrogate management framework. Optimization and Engineering 5, 235–262 (2004)