Chapter 2
Hybrid Optimization Algorithms and Hybrid Response Surfaces

George S. Dulikravich and Marcelo J. Colaço

Abstract In this paper we will present some hybrid methodologies applied to optimization of complex systems. The paper is divided in two parts. The first part presents several automatic switching concepts among constituent optimizers in hybrid optimization, where different heuristic and deterministic techniques are combined to speed up the optimization task. In the second part, several high dimensional response surface generation algorithms are presented, where some very basic hybridization concepts are introduced.

Keywords Multiobjective optimization · Response surfaces · Hybrid optimizers

2.1 Introduction

Design of complex nonlinear engineering systems usually requires a large computational effort in the case of simulation, or a large amount of human and experimental resources in the case of experiments. Multi-dimensional topology of the objective function space of such problems has multiple local minima and large domains of possible variations of the design variables search space. A typical approach to finding the global minimum is to start with a large search space utilizing an entire population of initial guesses and advancing them simultaneously using any of the evolutionary optimization algorithms. Once the search space has narrowed sufficiently, the search process is switched to a fast and accurate gradient-based search algorithm to converge on the minimum. However, this simplistic semi-manual approach to sequential

G.S. Dulikravich (✉)
Multidisciplinary, Inverse Design, Robust Optimization and Control (MAIDROC) Laboratory, Department of Mechanical and Materials Engineering, Florida International University, EC 3462, 10555 West Flagler St, Miami, FL 33174, USA
e-mail: dulikrav@fiu.edu

M.J. Colaço
Department of Mechanical Engineering, Biofuels and Energy Efficiency Program - Brazilian National Agency of Oil, Gas and Biofuels, Federal University of Rio de Janeiro, Cx. Postal 68503, Rio de Janeiro, RJ 21941-972, Brazil
e-mail: colaco@ufrj.br

D. Greiner et al. (eds.), *Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences*, Computational Methods in Applied Sciences 36, DOI 10.1007/978-3-319-11541-2_2

hybrid optimization is not reliable since it utilizes only one evolutionary optimizer and one gradient-based optimizer, each of which has its own intrinsic deficiencies. A more robust and faster hybrid optimization approach utilizes a collection of several evolutionary optimizers and several gradient-based optimizers and automatically switches among them. This chapter will focus on these types of hybrid optimizers.

When systems having large number of design variables, objective functions and constraints need to be optimized, this implies the evaluation of thousands and even millions of candidate solutions, which can make this task impossible from a very practical point of view, especially if each such high fidelity evaluation of the objective function is time-consuming or expensive. Thus, it is important to develop surrogate models, also called metamodels, which approximate the response of the original problem, but using a much simpler mathematical formulation. The objective of this chapter is to present several common response surface models existing in the literature, and some hybridization among them. Some hybrid optimizers are also presented, based on heuristic and deterministic methods, which take advantage of these hybrid response surface models to improve the efficiency of the optimization task of complex engineering systems.

2.2 Hybrid Optimization Algorithm Concepts

It is well known that each optimization algorithm provides a unique approach to optimization with varying degrees of convergence, reliability, and robustness at different stages during the iterative process. The "no free lunch theorem" states [1] that no individual optimization algorithm is better than all the other optimization algorithms for all classes of optimization problems (Fig. 2.1). A natural response to this problem is to use hybrid optimization algorithms that combine individual constituent optimization algorithms in a sequential or parallel manner so that resulting software can utilize the specific advantages of each constituent algorithm. That is, a variety of individual constituent optimization algorithms that rely on different principles of operation are combined in a hybrid optimization algorithm as subroutines where a set of specified heuristic measures of the iterative convergence process is used to perform automatic switching among the constituent algorithms. This allows for automatic use of the most appropriate constituent optimization algorithm at each step of the global iterative optimization process. The automatic back-and-forth switching [2] among the constituent optimization algorithms can also be viewed as an automatic backup strategy so that, if one optimization algorithm fails, another optimization algorithm can automatically take over.

The key to the success of this hybrid optimization concept is the automatic switching strategy [2, 3] among the constituent optimization algorithms. One of the early single-objective hybrid optimization algorithms [4, 5] had three gradient-based (Davidon-Fletcher-Powell algorithm, Sequential Quadratic Programming and quasi-Newton algorithm of Pshenichny-Danilin) and three non-gradient-based (Genetic Algorithm, Nelder-Mead simplex algorithm, and Differential Evolution

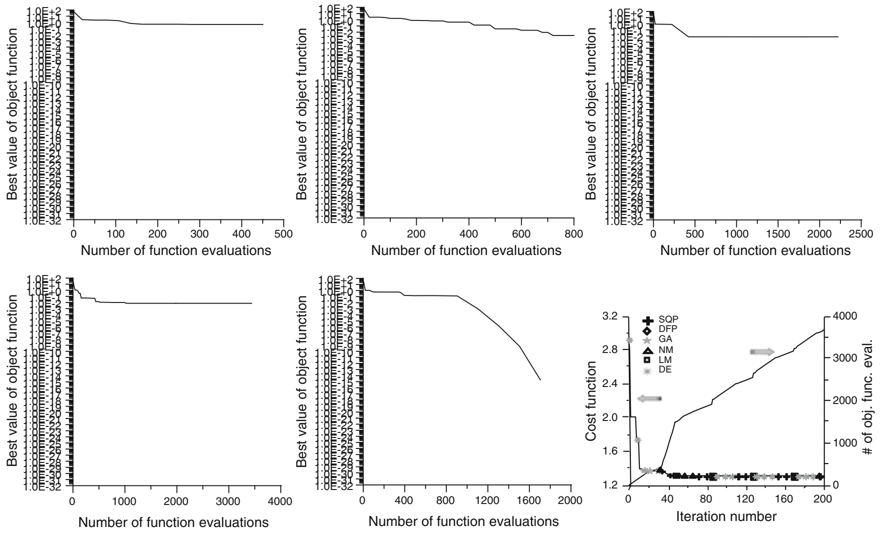

Fig. 2.1 Convergence histories for Griewank's test function no. 8 using (from *left* to *right*) individual: BFGS algorithm, differential evolution, simulated annealing, particle swarm, and our hybrid optimizer [27]. The last figure illustrates the convergence history of one hybrid optimizer using automatic switching process [5]

algorithm) constituent optimization algorithms that were automatically switching back-and-forth each time when a particular heuristic prescribed convergence measure was reached [5].

This hybrid single-objective optimizer only restarts with a single design (the "best" from the "previous" iteration). In other words, when switching from one of the population-based constituent optimizers to a gradient-based constituent optimizer, only the best design from that population, and not the entire population, is used as the initial guess for the gradient-based constituent algorithm.

For population-based constituent optimizers used in this hybrid optimizer, the population matrix was updated every iteration with new designs and ranked according to the value of the objective function. The optimization problem was completed when: (1) the maximum number of iterations or objective function evaluations was exceeded, or (2) the best design in the population was equivalent to a target design, or (3) the optimization program tried all four algorithms, but failed to produce a decrease in the objective function.

Another hybrid single-objective optimization algorithm was developed by combining three of the fastest gradient-based and evolutionary optimization algorithms [5], namely: the Broyden-Fletcher-Goldfarb-Shanno quasi-Newton algorithm, the Differential Evolution algorithm, and the Particle Swarm algorithm. It was found that the most often automatically used constituent optimization module is the Particle Swarm algorithm. When a certain percentage of the particles find a minimum, the algorithm switches automatically to the Differential Evolution algorithm and the particles are forced to breed. If there is an improvement in the objective function, the algorithm returns to the Particle Swarm method, meaning that some other region is more likely to have a global minimum. If there is no improvement in the objective function value, this can indicate that this region already contains the global value expected and the algorithm automatically switches to the Broyden-Fletcher-Goldfarb-Shanno algorithm in order to quickly and accurately find the location of the minimum.

One of the most recent switching algorithms is the Search Vector-based Hybrid (SVH) [2] which automatically changes search directions during the optimization process. During each iteration, the SVH will generate the SVs based on a predetermined formula or quality. Some examples of the SVs include:

1. Global Best vector (GB) which is the fittest design vector currently in the population.
2. Population Weighted Average vector (PWA): The population is ranked from best to worst, with the best receiving a rank equal to the population size, and the worst having a rank of one. The ranks are then used as weights, and the standard weighted arithmetic mean procedure is used to create this SV.

After the SVs have been evaluated, the fittest SV is selected as the SV for that iteration. Once the SV has been selected, the constituent algorithm selection process begins. First, each constituent algorithm is executed so that it generates a temporary small population. This temporary population will not be evaluated. Instead, it will be used as an indication of the behavior of the constituent algorithm for a given topology. For example, suppose the SVH has two constituent algorithms called CA1 and CA2 where CA1 will use the current population to generate a temporary population which will be situated in one part of the space of design variables, while CA2 will create a temporary population shifted to another part of the space of design variables. In order to select the most appropriate constituent optimization algorithm, the Euclidean distance between the endpoint of each centroid vector of the temporary populations and the selected SV is calculated and stored. Then, each centroid is evaluated. The constituent algorithms are then ranked using the Pareto dominance scheme based on two objectives: (1) minimize distance between the centroid and the SV, and (2) minimum objective function value of the centroid. The constituent algorithm to be used is randomly selected from the Pareto front. In order for the centroids of the temporary populations for the constituent algorithms to be statistically meaningful, the constituent algorithms are executed $10\times$ each iteration. Once a constituent algorithm has been selected, it is then executed one last time. This time, the population is permanently changed, and the objective function for each design vector is evaluated. This completes one full iteration of the SVH. This strategy [2] differs from any other known work in that it uses a collection of different search directions, each with its own unique formulation, and chooses among them. The method presented by Ahrari et al. [6], like most other hybrid algorithms, generates the search direction and keeps it fixed throughout the entire optimization process.

In multi-objective optimization it would be onerous to use a single value and compare the quality of one Pareto approximation to another [7]. Instead, multiple attributes of two successive Pareto approximations should be considered to determine if a multi-objective optimizer is converging on non-dominated set. Since “no free lunch theorem” applies to multi-objective optimizers as well as single-objective optimizers, hybrid optimization is highly advisable in the multi-objective optimization problems. Such hybridization can be implemented by using a suite of multi-objective optimization algorithms in the form of either High-level Relay Hybrid (HRH) algorithms where each of the constitutive algorithms run on its own in a sequential non-parallelized scheme, or as High-level Teamwork Hybrid (HTH) metaheuristic algorithms [8] where constitutive optimization algorithms run in parallel and contribute a portion of each new generation’s population. The portion that each search contributes to the new generation is dependent on the success of the algorithm to provide past useful solutions to the search. One such HTH algorithm is AMALGAM [9], which utilizes NSGA-II [7] and outperforms NSGA-II.

A robust and accurate HRH type concept is Multi-Objective Hybrid Optimizer (MOHO) [10] which currently uses three multi-objective optimization algorithms: Strength Pareto Evolutionary Algorithm (SPEA-2), a multi-objective implementation of the single objective Particle Swarm, and a Non-Sorting Differential Evolution (NSDE) algorithm which is a low level hybrid metaheuristic search combining NSGA-II [7] and Differential Evolution. MOHO starts by creating the population that will be used for the optimization run. The population contains the decision vector and the objective vector for all population points and stores the Pareto approximation and clustering routine. Clustering is performed by the population on the object vectors of the Pareto approximation. The decision vectors of all population points are evenly distributed over the decision space using Sobol’s pseudo random sequence generator [11]. The software then passes the population from optimization routine to optimization routine as the switching criteria dictates. The constitutive algorithm that is selected at each generation makes a new generation using any or all of the information provided to it: the last generation’s population and the latest non-dominated set. Then MOHO combines the new generation and the latest non-dominated set to create a new non-dominated set. The switching algorithm compares the non-dominated set from the current generation to the non-dominated set of the previous generation. The comparison process consists of looking at five desired improvements to the Pareto approximation [10]. The improvements are actually gains in five performance criteria (quality factors). If the particular search algorithm can achieve at least two of any of the five specified improvements [10], this algorithm is allowed to create the next generation. The five criteria (aspects) are:

1. The new population changes the number of points in the Pareto set. When this happens, either points are being added to the approximation, or, more importantly, a new point is found that causes points to be deleted from the Pareto set.
2. The new population has at least one point that dominates a point, or points, in the current Pareto approximation. This means that the Pareto approximation is being improved.
3. The hyper volume of the dominated space changes. When the optimization software starts, it picks a worst case objective vector from its initial population guess.

At each constituent optimization algorithm iteration, hyper cubes are created, with one vertex of the diagonal being a point on Pareto approximation and the other vertex of the diagonal being the worst case objective vector. Then the volume of the union of all the hyper cubes is calculated. The union is defined as the Boolean union of the cubes; in the same sense as this operation is performed in Constructive Solid Geometry for CAD applications. When this occurs, the Pareto approximation is changing geometry.

4. The new population generation causes the average distance of Pareto approximation from the objective space origin to change. This also denotes a change on the Pareto approximation geometry. This is a backup to criterion (3) where two approximations may have the same volume, but different average distances.
5. The new population causes the maximum spread of the Pareto approximation to increase. The formula for calculating the spread developed by Zitzler is shown by Deb [7].

At the end of each iteration, the population of design vectors assigns itself a grade point for each of the above criteria that its new generation meets. If the new population earns a grade of 2 or more, the current optimization routine is allowed to continue running. When the grade falls below 2, the software switches to the next optimizer in its repository. If the grade is 0 or 1, the reason to switch to another constituent optimizer is because the currently used constituent algorithm is not contributing to improving the Pareto approximation. As an example, the population gives itself a grade of 1 because it meets criterion 4. This change could be caused just by clustering of the previous Pareto approximation and the new population. While this type of change in the Pareto approximation has its uses, it has been found that the multi-objective routines used here can cause these kind of changes to the Pareto approximation *ad infinitum*, when the Pareto approximation is very near the actual non-dominated set of the objective space. It has been found that by enforcing at least 2 of the criteria, these situations are avoided.

The other limiting factor on how many consecutive iterations a given optimization routine can run is the sub-iteration limit. Although a routine may be able to score a grade of at least 2 indefinitely, for each new generation, there may be an optimization method available that can do a better job. For this reason, each constituent optimizer is limited to a user defined maximum sub-iteration limit. This limit gives all the constituent optimizers a chance to run.

2.3 Hybrid Response Surface Generation Concepts

Optimization of systems with a large number of design variables where the objective function is given in a pointwise fashion requires creation of a hypersurface that fits through the given values of the objective function in a multi-dimensional space where dimensionality corresponds to the number of the design variables. It is well known that the locations of the training points are crucial for the proper construction of the response surface. When we are given a freedom to choose the locations of the support points of a multi-dimensional response surface, a typical approach is to use *Design of Experiments* (DoE) for this purpose. For high dimensional problems, strategies such as Latin Hypercube Sampling [11], Sobol quasi-random sequences of numbers [12], and a variety of *de facto* random number generators are most often used. However, when we do not have freedom to choose the number and the locations of the support points for generation of a response surface, all existing methods for generating response surfaces have serious problems with accuracy and robustness. This is mainly because arbitrary data sets provide inadequate uniformity of coverage of space of the design variables and clustering of the support points that leads to spurious oscillations of the response surfaces.

2.3.1 Polynomial Regression

The use of polynomial regression is one of the earliest attempts to generate response surface models [13, 14]. The idea is to approximate an unknown function $f(\mathbf{x})$ by an approximation $s(\mathbf{x})$ in the following general form

$$\begin{aligned} f(\mathbf{x}) \approx s(\mathbf{x}) = a_0 &+ \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n}\sum_{j\le i}^{n} a_{ij}x_i x_j \\ &+ \sum_{i=1}^{n}\sum_{j\le i}^{n}\sum_{k\le j}^{n} a_{ijk}x_i x_j x_k + \cdots \end{aligned} \tag{2.1}$$

where n is the number of dimensions of the problem. Notice that we can write Eq. (2.1) as

$$\mathbf{f} = \mathbf{X}\mathbf{a} + \varepsilon \tag{2.2}$$

where ε is an approximation error with zero mean and variance, σ^2. If the functions are given at certain known locations, the unknowns $\mathbf{a}$ can be found by least squares

$$\mathbf{a} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{f} \tag{2.3}$$

It is well known that the locations of the training points are crucial for the proper construction of the response surface. Such choice is known in the literature as *Design of Experiments* (DoE). For low dimension problems, the classical method of choice was the factorial design [13], which is not practical for high dimensional problems. Other strategies for high dimensions include the Latin Hypercube Sampling [11], Sobol quasi-random numbers [12], etc.

Although the polynomial regression technique seems very attractive in view of its simplicity, it is not practical when the number of dimensions of the problem becomes very high. In this case, there are other relatively recent techniques more appropriate.

Among those techniques, the most used are Kriging [15], Radial Basis Functions [16], and Neural Networks [17], among others.

2.3.2 Self Organizing Algorithms [19, 20]

Self-organizing algorithms come from the field of cybernetics [18]. The idea is that the program "learns" the black box model as it is trying to mimic and makes the response surface as complex as is required. By allowing the program to gradually complicate the final model, the construction and evaluation time of the surrogate model is automatically optimized for a given task. The black box model is the test function being used to evaluate the RSM method.

The self-organizing algorithm presented here [19], is the multilayer algorithm where the design variables are permutated, in pairs, to form nodes. At each node a least squares regression is performed using the two variables input to the nodes. These are variable vectors that are the size of the sample population. Thus, the output of the node is a vector of the predicted values from the regression. The polynomial used for the regression is a first order or second order polynomial. For instance, a second order basis polynomial would be:

$$\begin{aligned} y_{i,j}^{k,n} &= a_0^{k,n} + a_1{}^{k,n} x_i^{k,n} + a_2{}^{k,n} x_j^{k,n} + a_3{}^{k,n} x_i^{k,n} x_j^{k,n} \\ &\quad + a_4{}^{k,n} x_i^{k,n} x_i^{k,n} + a_5{}^{k,n} x_j^{k,n} x_j^{k,n} \quad i \neq j \end{aligned} \tag{2.4}$$

where $i, j = 1, 2, \ldots,$ number of inputs to a given layer, k is the current layer, and n is the node number at current layer. Equation (2.4) would then be a row in the Vander Monde matrix for a regression using a second order basis polynomial of two variables. The output of the node is the vector of predicted y values for the given input. The output of a node in layer $k-1$ becomes the input (provides an x_i vector) for layer k.

The notation in Eq. (2.4) is designed to inform the reader that the functions and polynomial coefficients pertain to a particular layer and particular node in the layer. The notation should also give the reader a feel for the computational resources needed to create and maintain a multilayer self-organizing model.

Figure 2.2 shows a possible multilayer surrogate model for a three variable engineering model. In the bottom layer (the zero layer) actual design variables are the nodes in the layer. These become the x inputs to layer 1. The nodes for layer 1 are created by permuting the input variables and performing a least squares fit using Eq. (2.4), and the actual responses from the actual function (that is, objective function, engineering simulation, etc.). Once layer 1 is made, layer 2 is created, but now the nodes of layer 1 provide the x's to make the new nodes using Eq. (2.4). When layer 3 is to be created, the 3rd node of layer 2 is not included. For now, we will just say that the results of that regression were not good enough to be used to make the 3rd layer. Since only two nodes from layer 2 were used to make layer 3, only one node can be created in layer 3 and the model making process ends there.

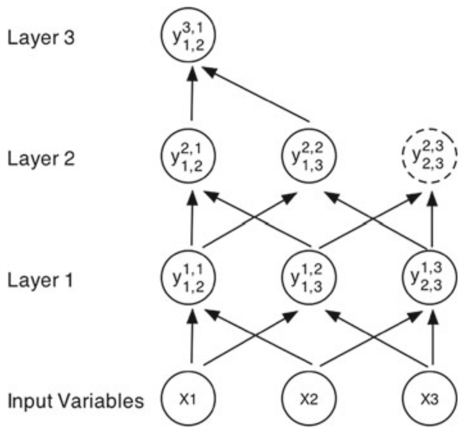

Fig. 2.2 Sample multilayer model for self-organizing response surface creation [19]

The third node in layer 2 is not used to make nodes in layer 3 in Fig. 2.2. A selection criterion is used to determine if the information in a node gets passed on to the next layer. Madalaand and Ivakhnenko [18] suggest that using the following equation is an appropriate means for checking the quality of a node and can be a selection criterion

$$\Delta^2(B) = \frac{\sum_{p \in N_B} (y - \widehat{y})_p^2}{\sum_{p \in N_B} (y_p - \bar{y})^2} \tag{2.5}$$

where y_p are the desired values, $\widehat{y}$ are the predicted values and $\bar{y}$ are the mean of the desired values. In the multilayer algorithm a threshold is set for the maximum acceptable value of Eq. (2.5). Nodes that are within the threshold are passed on to the next layer. For each new layer the threshold is made smaller. This serves to minimize the amount of nodes in each layer to only the information that is needed to improve the network. This trimming of nodes is crucial to keeping the method compact and efficient [19]. The reader is urged to follow the example in Fig. 2.2, but first setting the number of input variables to 4. The rate of the growth of layers can be very large.

The building of the multilayer network can be terminated in two ways (in practice): A) Build a predetermined number of layers and chose the node in the last layer with the best value of Eq. (2.5) to be the model output. B) Build layers until all nodes are unable to meet the threshold value, chose the best-valued node as the output of the model. Once the output node is chosen, the polynomial coefficients pertaining to all the nodes used to create the output node are stored for evaluation of the model (extraction of a predicted value).

2.3.3 Kriging

The Kriging technique was named after the initial work of the South African mining engineer Krige [20]. Details of this technique can be found in the classical papers of Sacks et al. [21, 22] and Matheron [15]. It is worth mentioning also the excellent work of Jones et al. [23] where they proposed an efficient method to global optimization. Later, Huang et al. [24] extended Jones et al. work to deal with model uncertainties. Following procedure described in the works of Jones et al. [23] and Sacks et al. [21, 22], the Kriging method starts by constructing a stochastic process model for the function as

$$f\left(\mathbf{x}^i\right) = \sum_{j=1}^{m} a_j g_j\left(\mathbf{x}^i\right) + \varepsilon^i \tag{2.6}$$

where the superscript i is used to denote the ith point in the space of design variables $\mathbf{x}$. In Eq. (2.6), g_j is a set of m linear or non-linear functions and ε is an approximation error with zero mean and variance, σ^2. In the classical Kriging model, the approximation error is supposed to be function of the design variables, such that $\varepsilon^i = \varepsilon(\mathbf{x}^i)$. Some recent works [24] also include a measurement error in Eq. (2.6), but this will not be discussed here. A usual hypothesis in this model is that if two points $\mathbf{x}^i$ and $\mathbf{x}^j$ are close, then their approximation errors $\varepsilon(\mathbf{x}^i)$ and $\varepsilon(\mathbf{x}^j)$ are also close, meaning that $\varepsilon(\mathbf{x}^i)$ and $\varepsilon(\mathbf{x}^j)$ are correlated. The correlation function between those two errors can be given as a function of the weighted distance between then [21, 22].

$$d\left(\mathbf{x}^i, \mathbf{x}^j\right) = \sum_{k=1}^{n} \theta_k \left|x_k^i - x_k^j\right|^{P_k} \qquad \theta_k \geq 0,\ P_k \in [1, 2] \tag{2.7}$$

where n is the number of dimensions of the problem. The correlation is given as

$$\mathrm{Corr}\left[\varepsilon\left(\mathbf{x}^i\right), \varepsilon\left(\mathbf{x}^j\right)\right] = \exp\left[-d\left(\mathbf{x}^i, \mathbf{x}^j\right)\right] \tag{2.8}$$

According to Jones et al. [23], such model is so powerful that we can rewrite Eq. (2.6) in terms of the mean of the stochastic process μ, such that

$$f\left(\mathbf{x}^i\right) = \mu + \varepsilon\left(\mathbf{x}^i\right) \tag{2.9}$$

Following Jones et al. [23] we can then obtain the approximate function for a new point $\mathbf{x}^*$ as

$$f\left(\mathbf{x}^*\right) = \hat{\mu} + \mathbf{r}^T \mathbf{R}^{-1}\left(\mathbf{f} - \mathbf{1}\hat{\mu}\right) \tag{2.10}$$

where $\mathbf{1}$ is vector of ones, $R_{i,j} = \mathrm{Corr}[\varepsilon(\mathbf{x}^i), \varepsilon(\mathbf{x}^j)]$, $r_i = \mathrm{Corr}[\varepsilon(\mathbf{x}^*), \varepsilon(\mathbf{x}^i)]$, $\mathbf{f}$ is the vector of functions at the known locations, and

$$\hat{\mu} = \frac{\mathbf{1}\mathbf{R}^{-1}\mathbf{f}}{\mathbf{1}^T\mathbf{R}^{-1}\mathbf{1}} \tag{2.11}$$

Kriging also predicts the mean squared error of the estimates [21–23] and this has been used as a predictor to locations where to add points in the response surface model. Locations of the domain where the mean squared error of the estimates are large, usually require the addition of extra points to increase the local accuracy.

2.3.4 Radial Basis Functions

The use of Radial Basis Functions (RBFs), initially proposed in the work of Hardy [25] on multivariate approximation, is now becoming an established approach. RBFs may be classified into two main groups:

- The globally supported ones namely the multiquadrics (MQ, $\sqrt{\left(x-x_j\right)^2+c_j^2}$, where c_j is a shape parameter), the inverse multiquadrics, thin plate splines, Gaussians, etc;
- The compactly supported ones such as the Wendland [26] family (for example, $(1-r)_+^n + p\,(r)$ where $p(r)$ is a polynomial and $(1-r)_+^n$ is 0 for r greater than the support).

Let us suppose that we have a function of L variables x_i, $i = 1, \ldots, L$. One possible RBF approximation [27] can be written as

$$f\,(\mathbf{x}) \approx s\,(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \phi\left(\left|\mathbf{x}-\mathbf{x}_j\right|\right) + \sum_{k=1}^{M}\sum_{i=1}^{L} \beta_{i,k} p_k\,(x_i) + \beta_0 \tag{2.12}$$

where $\mathbf{x} = \{x_1, \ldots, x_i, \ldots, x_L\}$ and $f(\mathbf{x})$ is known for a series of points $\mathbf{x}$. Here, $p_k(x_i)$ is one of the M terms of a given basis of polynomials [28]. This approximation is solved for the α_j and $\beta_{i,k}$ unknowns from the system of N linear equations, subject to the conditions (for the sake of uniqueness)

$$\begin{gathered} \sum_{j=1}^{N} \alpha_j p_k\,(x_i) = 0 \\ \vdots \\ \sum_{j=1}^{N} \alpha_j p_k\,(x_L) = 0 \end{gathered} \tag{2.13}$$

$$\sum_{j=1}^{N} \alpha_j = 0 \tag{2.14}$$

One of the possible RBFs are the multiquadrics radial functions

$$\phi\left(\left|\mathbf{x}_i-\mathbf{x}_j\right|\right)=\sqrt{\left(\mathbf{x}_i-\mathbf{x}_j\right)^2+c_j^2} \tag{2.15}$$

where the shape parameter c_j must be adjusted. According to Baxter [29], usually large values of c_j provide the best approximations.

2.3.5 Wavelet Based Neural Networks [31, 32]

Wavelets occur in the family of functions generated from a mother wavelet $\psi(x)$. Each wavelet in it is defined by dilatation vector, a_i, which controls the scaling, and translation vector, t_i, which controls the position. Given a training set, the overall response of a WNN can be arithmetically written as

$$s\left(\mathbf{x}\right)=W_0+\sum_{i=1}^{N_p}W_i\psi_i\left(\frac{\bar{x}-\bar{t}}{\bar{a}}\right) \tag{2.16}$$

where N_P is the number of wavelet nodes in the hidden layer and w_i is the synaptic weight for each hidden node in the WNN. The dilatation and translation vectors have size equal to the number of variables in the estimated function. Such a network can be used to approximate any function

$$f\left(\mathbf{x}\right)=s\left(\mathbf{x}\right)+\varepsilon \tag{2.17}$$

where s is a regression function and the error term ε is a zero-mean random variable of disturbance.

One of the well known approaches for constructing WNN [30] requires the generation of a wavelet library, W. This library is composed of discretely dilatated and translated versions of mother wavelet function, $\psi\left(x\right)$. The next step is selecting the best wavelets based on the training data from this library to build the regression. This approach for building WNN becomes prohibitively computationally expensive when the estimated function has a large number of variables. This is due to exponential increase of the size of the wavelet library W with the dimension of the estimated function. Searching such a huge library one-by-one is computationally redundant. Therefore, a stochastic approach should be used for searching the best wavelets for the WNN hidden nodes [31].

2.4 Hybrid Methods for Response Surfaces

In this section we will present some hybrid response surface methods. The accuracy of these methods, along with their comparison against other strategies, will be presented in the next section.

2.4.1 Fittest Polynomial Radial Basis Function (FP-RBF) [28]

The FP-RBF hybrid method [27] consists of choosing the best possible combination of RBF, polynomial order, variable scaling, and shape parameter for a given problem.

In the work of Colaço et al. [27], the polynomial part of Eq. (2.12) was taken as

$$p_k\left(x_i\right) = x_i^k \tag{2.18}$$

and the radial basis functions were selected among the following

$$\text{Multiquadrics:}\quad \phi\left(\left|x_i - x_j\right|\right) = \sqrt{\left(x_i - x_j\right)^2 + c_j^2} \tag{2.19}$$

$$\text{Gaussian:}\quad \phi\left(\left|x_i - x_j\right|\right) = \exp\left[-c_j^2\left(x_i - x_j\right)^2\right] \tag{2.20}$$

$$\text{Squared multiquadrics:}\quad \phi\left(\left|x_i - x_j\right|\right) = \left(x_i - x_j\right)^2 + c_j^2 \tag{2.21}$$

$$\text{Cubical multiquadrics:}\quad \phi\left(\left|x_i - x_j\right|\right) = \left[\sqrt{\left(x_i - x_j\right)^2 + c_j^2}\right]^3 \tag{2.22}$$

Some tests were made using the cross-product polynomials ($x_i x_j x_k \ldots$), but the improvements on the results were found out to be irrelevant [27]. Also, other types of RBFs were previously considered by the authors [27], but no improvement in the accuracy of the interpolation was observed.

Therefore, a polynomial of order M is added to the radial basis function. After inspecting Eqs. (2.12)–(2.14), (2.18), one can easily check that the final linear system has [(N+M*L)+1] equations that can be solved by any traditional technique.

In the technique presented by Colaco et al. [27], initially all variables have to be normalized. Then, the initial guess for the shape parameter c is set as the minimum distance between two points in the training set of variables. Shape parameter, c, is then increased until the best solution is obtained. Also, different scaling of the variables are tried to give the best fit for the function. The choice of which polynomial order, which shape parameter and scaling of the variables, and which RBF are the best for fitting a specific data set was made based on a cross-validation procedure. Let us suppose that we have PTR training points, which are the locations in the multidimensional space where the values of the function are known. Such set of training points is equally subdivided into two subsets of points, named PTR1 and PTR2. The Eqs. (2.12)–(2.14) are solved for a polynomial of order zero and for one of the RBF expression given by Eqs. (2.19)–(2.22) using the subset PTR1. Then, the value of the interpolated function is checked against the known values of the function that are in the subset PTR2. The error is recorded as

$$RMS_{PTR1,M=0,RBF1} = \sum_{i=1}^{P_{TR2}} \left[s(\mathbf{x}_i) - f(\mathbf{x}_i)\right]^2 \tag{2.23}$$

Then, the same procedure is repeated by using the subset PTR2 to solve the equations and the subset PTR1 to calculate the error as

$$RMS_{PTR2,M=0,RBF1} = \sum_{i=1}^{P_{TR1}} \left[s(\mathbf{x}_i) - f(\mathbf{x}_i) \right]^2 \tag{2.24}$$

Finally, the total error for the polynomial of order zero using one of the RBF expressions given by Eqs. (2.19)–(2.22) is obtained as

$$RMS_{M=0,RBF1} = \sqrt{RMS_{PTR1,m=0,RBF1} + RMS_{PTR2,m=0,RBF1}} \tag{2.25}$$

This procedure is repeated for all polynomial orders, up to $M = 10$ and for each one of the RBF expressions given by Eqs. (2.19)–(2.22). The best combination is the one that returns the lowest value of the RMS error. Although this cross-validation procedure is quite simple, it worked very well for practical test cases including optimization of chemical compositions of alloys [32, 33] ,maghnetohydrodynamic flow in cavities [34], energy/exergy optimization [35, 36] and Bayesian inverse problems in heat transfer [37].

2.4.2 Kriging Approximation with Fittest Polynomial Radial Basis Function (KRG-FP-RBF)

A new method is proposed in this paper, which combines the very high accuracy of the FP-RBF approximation with the stochastic appealing of the Kriging method. The idea is conceptually simple, although the computational implementation requires some effort. Results presented in this paper are still preliminary and need further investigation.

Referring to Jones et al. [23], the Kriging approximation given by Eq. (2.6) is so powerful that the base function $g(\mathbf{x})$ can be written as the mean of the stochastic process, μ. However, we propose to extend even more such accuracy by using the FP-RBF approximation as a base function for the Kriging process. In other words, once the FP-RBF method has been adjusted and fitted to a particular set of data, we use this approximation as the $g(\mathbf{x})$ function in the Kriging model given by Eq. (2.6). Inserting the FP-RBF definition given by Eq. (2.12) into the Kriging model of Eq. (2.6), we have the following stochastic process model for the function as

$$f\left(\mathbf{x}^i\right) = a\, s\left(\mathbf{x}^i\right) + \varepsilon^i \tag{2.26}$$

Notice that $s(\mathbf{x})$ has to be built using the FP-RBF model explained earlier. By doing this, we are assuming that ε is the FP-RBF approximation error, which has (by hypothesis) zero mean and variance σ^2. Once the approximation was built by the FP-RBF method, the Kriging approximation is used to model such error in a stochastic way. Then, following Sacks et al. [21, 22] derivations, adapted to our nomenclature, we can obtain the best linear unbiased predictor at a new point $\mathbf{x}^*$ as (mathematical details are omitted for lack of space, but the interested readers are urged to read publications by Sacks et al. [21, 22] for all details of this derivation)

$$\tilde{s}\left(\mathbf{x}^*\right) = a\, s\left(\mathbf{x}^*\right) + \mathbf{r}^T\left(\mathbf{x}^*\right)\mathbf{R}^{-1}\left[\mathbf{F} - a\,\mathbf{S}\right] \tag{2.27}$$

where $\mathbf{F}$ is the vector of the exact function evaluated at the training points locations and $\mathbf{S}$ is vector of FP-RBF approximation at these same locations. Notice that if $a = 1$, and $\mathbf{F} = \mathbf{S}$, then the Kriging model reduces to the FP-RBF model. Thus, the second term gives some measure of the error between the real data and the FP-RBF approximation. In this equation, $\mathbf{R}$ is the full correlation matrix between the training points, given by Eq. (2.8) and $\mathbf{r}$ is the correlation vector between the evaluation point $\mathbf{x}^*$ and the training points, also given by Eq. (2.8). The parameters θ and P, appearing in Eq. (2.7) can be obtained by minimizing $[(\det\mathbf{R})^{1/m}\sigma^2]$ where m is the number of training points [21, 22]. In this paper, such minimization was performed by the Particle Swarm method. The other parameters appearing in these equations are given as [21, 22]

$$\sigma^2 = \frac{1}{m}\left(\mathbf{F} - a\,\mathbf{S}\right)^T\mathbf{R}^{-1}\left(\mathbf{F} - a\,\mathbf{S}\right) \tag{2.28}$$

$$a = \left(\mathbf{S}^T\mathbf{R}^{-1}\mathbf{S}\right)^{-1}\mathbf{S}^T\mathbf{R}^{-1}\mathbf{F} \tag{2.29}$$

where Eq. (2.28) gives the maximum likelihood estimation of the variance and Eq. (2.29) is the generalized least-squares estimate [21, 22] of a. Thus, this procedure uses the Kriging method to model the approximation error of the FP-RBF approximation.

2.4.3 Hybrid Self Organizing Model With RBF [20]

The best known application of self-organizing method is in the commercial software IOSO [38] which uses quadratic local fitting polynomials. A more general idea is to use the self-organizing method given by Eq. (2.4) to choose the best local fitting functions (linear, quadratic, cubic or quadratic) to generate a response surface thus capturing the major topology of the response multi-dimensional hyper-surface. However, this metamodel does not force the hyper-surface to pass exactly through the provided support points. The difference between the actual values of the objective function at the support points and the fitted values at the support points represents a much less challenging topology which it then fitted using RBF method [19]. Such a combined response surface fitting method is much more robust and accurate (Figs. 2.3 and 2.4) than either of the separate methods used in this hybrid [19]. This hybrid

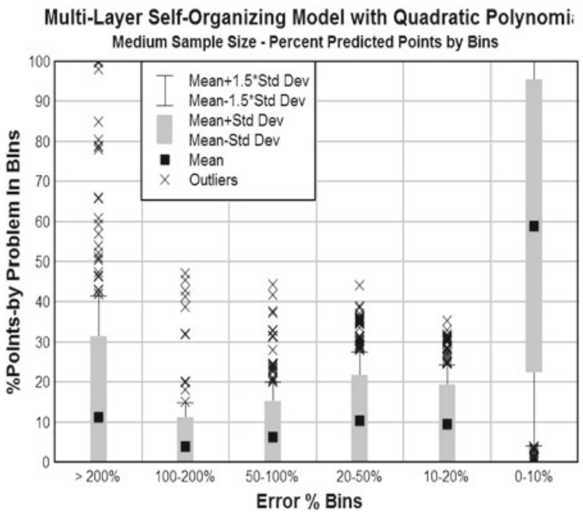

Fig. 2.3 Self-organizing method using quadratic local polynomials applied to a medium size data set; model accuracy [19]

method is simple to implement, although computationally costly, where computing cost is mainly due to the self-organizing method. This suggests a need for research into reducing the computing cost of the self-organizing method by using more efficient pruning algorithms to eliminate those branches in the genetic tree that are terminating.

2.4.4 Genetic Algorithm Based Wavelet Neural Network (HYBWNN) [31, 32]

Another hybrid method used for fitting high dimensional functions is the Genetic Algorithm Wavelet Based Neural Network (WNN) model presented by Sahoo and Dulikravich [31] with 5 neural subnets. Typically, the mother wavelet used in the WNN is Mexican Hat wavelet given by

$$\psi(x) = \left(\frac{2}{\sqrt{3}}\pi^{-1/4}\right)\left(1 - x^2\right)\exp\left(\frac{-x^2}{2}\right) \tag{2.30}$$

Gaussian wavelets were also used along with this mother wavelet to construct the WNN [31]. For each node of the WNN, genetic algorithm was used to search the best Mexican Hat wavelet and the best Gaussian wavelet. The one having a lower norm of residue after performing multiple linear regression was selected and used in the WNN architecture. The concept of binary genetic algorithm was used to

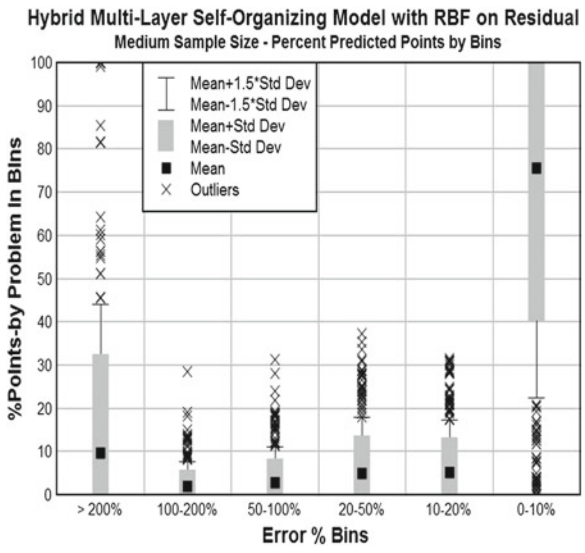

Fig. 2.4 Self-organizing method followed by RBF used on residuals applied to a medium size data set; model accuracy [19]

t1					a1			t2					a2		
1	0	1	1	1	1	1	1	1	1	0	1	0	0	0	1

Fig. 2.5 Binary string representation of a 2-D wavelet

search for the wavelets required for the hidden nodes in the WNN. The dilatation and translation factors (binary representation) in Eq. (2.16) for each dimension of the wavelet were concatenated to form the chromosomes in the GA population. A typical representation of a wavelet in 2-D function estimation is shown in Fig. 2.5.

It has two dilatation factors specifying the scaling and two translation factors specifying the position of the wavelet in each dimension. The variables space is normalized so the translation factors can vary within $[-1, 1]$ and the dilatation factors can vary within [0.1, 0.8]. The fitness for selecting the wavelet was defined as the norm of the residue obtained by doing multiple linear regression of the values given by the wavelet transform of the training data versus the real function values. The GA was run for a sufficient number of generations to select a wavelet. Subsequent wavelets were searched by the GA based on the residue obtained in former step set as target values. This approach was unable to search for proper wavelets when the number of variables in the estimated function went beyond ten. The chromosome length for such functions was huge and the binary GA became inefficient. Therefore, a real numbers GA search was proposed where the wavelet is represented as a string of real number instead of a binary string. The range for searching for the values of dilation factors was relaxed to [0.005, 5.00]. This gave more flexibility to the GA for

a1	t1	a2	t2	a3	t3	a4	t4
2.8424	0.25483	2.4672	0.000922	1.7407	0.98521	2.5848	-0.61214

Fig. 2.6 Real string representation of a 4-D wavelet

searching appropriate wavelets [31]. A typical example of a wavelet representation in 4-D function estimation is shown in Fig. 2.6.

The fitness assignment was similar to the previous method. In addition, whole arithmetic crossover and floating point mutation operators were used. Separate GAs were run serially [36] for finding the activation function in each node of the WNN architecture.

2.5 Comparison Among Different Response Surface Algorithms

Performance of different hybrid response surface algorithms was evaluated on data sets containing either scarce ($3L$), small ($10L$), medium ($30L$) or large ($50L$) number of points, where L designates the dimensionality of the problem [39, 40].

2.5.1 Fittest Polynomial RBF Versus Hybrid Wavelet Neural Network [42]

In order to compare the accuracy of the FP-RBF [27] model against the Hybrid Wavelet Neural Network (WNN) [31], 13 test cases were used, representing linear and non-linear problems with up to 16 variables. These test cases, defined as problems 1–13 were selected by Jin et al. [41] in a comparative study among different kinds of meta-models. Such problems were selected from a collection of 395 problems (actually 296 test cases), proposed by Hock and Schittkowski [42] and Schittkowski [43]. For the other comparison presented in this paper, all 296 test cases will be presented. The reason is the very high computational cost associated with the WNN method that restricted us to these 13 test cases initially. Also, for these test cases, the polynomial degree of the FP-RBF model was fixed in a pre-specified value, and the shape parameter was set to $1/N$, where N is the number of training points. For the other sections of this paper, those two parameters were allowed to vary according to the cross-validation procedure defined previously.

The first 12 problems do not have random errors added to the original function, while the problem no. 13 has a noise added with the following form

$$\varepsilon(x_1, x_2) = \sigma r \tag{2.31}$$

where σ is the standard deviation and r is a random number with Gaussian distribution and zero mean.

For accuracy, the goodness of fit obtained from "training" data is not sufficient to assess the accuracy of newly predicted points. For this reason, additional confirmation samples are used to verify the accuracy of the metamodels. To provide a more complete picture of metamodel accuracy, two different metrics are used: R Square, and Relative Average Absolute Error (RAAE) [41].

(a) R Square (R2)

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n} \left(y_i - \bar{y}\right)^2} = 1 - \frac{MSE}{\text{variance}} \tag{2.32}$$

where $\hat{y}_i$ is the corresponding predicted value for the observed value y_i; $\bar{y}$ is the mean of the observed values. While *MSE* (Mean Square Error) represents the departure of the metamodel from the real simulation model, the variance captures how irregular the problem is. The larger the value of R2, the more accurate the metamodel.

(b) Relative Average Absolute Error (RAAE)

$$RAAE = \frac{\sum_{i=1}^{n} \left|y_i - \hat{y}_i\right|}{n^* STD} \tag{2.33}$$

where *STD* stands for standard deviation. The smaller the value of RAAE, the more accurate the metamodel.

The FP-RBF model presented here was compared against the WNN method for the 13 selected analytical test cases. In order to check the accuracy of the metamodel when different samples were employed, three different sets of training points were used, as suggested by Jin et al. [41]. Table 2.1 gives the number of training points, testing points, minimum and maximum value of each test function, as well as the standard deviation and average value of each test function.

Initially, the results obtained with the FP-RBF model, with a polynomial of order 10 using a *large* number of training points and the results obtained with a polynomial of order 1 for small and scarce sets of training points were compared with the results obtained by using WNN method [44]. Only problems no. 1–5 were tested for a scarce set of training points, as suggested by Jin et al. [41].

Figure 2.7 demonstrates that when considering R2 metric, for large and small sets of training points, the RBF was better than the WNN, while for a scarce number of training points, the WNN was more accurate. On the contrary, it appears that when considering RAAE metric, for large and small sets of training points, the WNN was better than the RBF, while for a scarce number of training points, the RBF was more accurate.

Table 2.1 Parameters for the 13 analytical test functions used by Jin et al. [41]

		Number of training and testing points				Non Linearity	Minimum value of f	Maximum value of f	Standard deviation of f	Average value of f
		Training			Testing					
PB #	Vars	Scarce	Small	Large						
PB1	10	30	100	198	1,000	High	−21.01	29.25	8.92	8.60
PB2	10	30	100	198		Low	−1717.43	−539.36	187.40	−1146.82
PB3	10	30	100	198		High	−1327241.16	−1.14	224561.73	−213124.84
PB4	10	30	100	198		Low	258.78	4779.01	1002.27	2185.25
PB5	16	48	160	459		Low	7136.99	195608.81	27294.72	69060.73
PB6	2	N/A	9	100		High	98.94	187.53	17.75	127.73
PB7	2		9	100		High	−0.26	0.26	0.15	0.00
PB8	2		9	100		Low	−1.51	8.91	2.08	1.66
PB9	3		27	125		High	2.15	146369.38	21527.31	10842.11
PB10	3		27	125		High	3.42	32.84	7.70	29.13
PB11	3		27	125		Low	40829.97	−40749.42	21.60	−40790.41
PB12	2		9	100		Low	−12.72	13.75	5.74	0.52
PB13	2		9	100		Low	−54.77	77.80	30.94	12.60

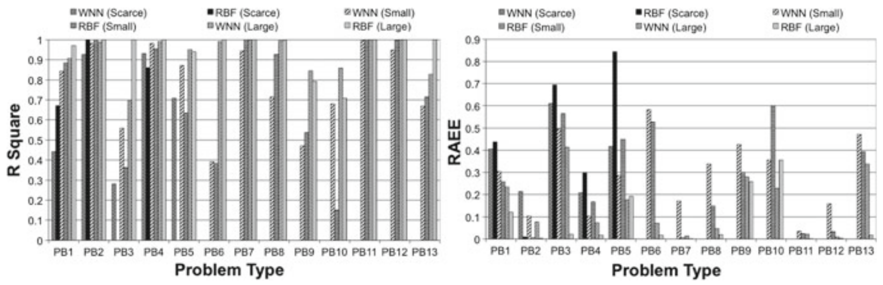

Fig. 2.7 R2 and RAAE metrics for WNN and FP-RBF

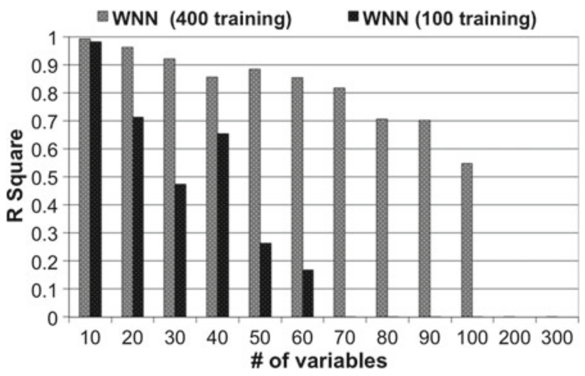

Fig. 2.8 R2 results with WNN for a large number of variables (WNN with five subnets)

However, one of the major problems with WNN is its rapidly decreasing accuracy with increasing dimensionality of the problem. Figure 2.8 demonstrates the results for test problem no. 2 when using the WNN. One can see that the accuracy, given by the R2 metric, decreases rapidly when using 100 training points. Also, for 400 training points, the R2 goes to a negative value when using more than 100 variables. Colaço et al. [39] demonstrated that the RBF model was able to maintain a very high accuracy even when the number of variables increased to 500.

Figure 2.9a shows the computational time required to run this test case using the FP-RBF model. The code was written in Fortran 90 and the CPU was an Intel T2300 1.66 Ghz (Centrino Duo) with 1 Gb RAM. Figure 2.9b shows the computational time required by the WNN where one can notice the extremely high computational cost. The code for the WNN was written in Matlab 7.0.4 and the CPU was an Intel T2300 1.66 Ghz (Centrino Duo) with 1 Gb RAM. Some improvement in the performance could be expected by converting this code to Fortran90 or C++. However, the computational cost for the WNN for a problem with 300 variables and 400 training points, even with different programming languages (Matlab and Fortran90) was approximately 6,000 times greater than for the RBF.

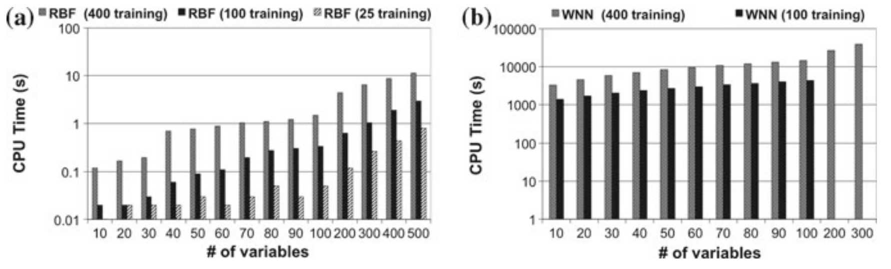

Fig. 2.9 Computing time for a large number of variables: (**a**) RBF with M = 1, (**b**) WNN with five subnets

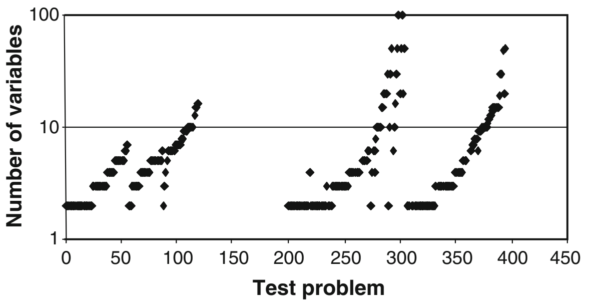

Fig. 2.10 Number of variables for each problem considered in the Schittkowski suite of analytical test cases [42, 43]

Table 2.2 Number of training and testing points

	Number of training points (L)	Number of testing points (L)
Scarce set	3	300
Small set	10	1,000

2.5.2 *Fittest Polynomial RBF Versus Kriging*

In this section we will compare the accuracy and computing time requirement of the FP-RBF model against the one given by the Kriging model proposed by Jones et al. [23]. From now on, 296 test cases will be used, representing linear and non-linear analytical problems with up to 100 variables. These test problems were selected from a collection of 395 (actually 296) test cases proposed by Hock and Schittkowski [42] and Schittkowski [43]. Figure 2.10 shows the number of variables of each test case analyzed. To verify the accuracy of the interpolation over different number of training points, two sets were defined. Also, the number of testing points varied, according to the number of training points. Table 2.2 presents these two sets, based on the number of dimensions (variables) *L* of the problem.

Initially, Fig. 2.11 presents the values of R2 and RAAE metrices for the FP-RBF model, considering a scarce set of data. As one can see, most of the test cases

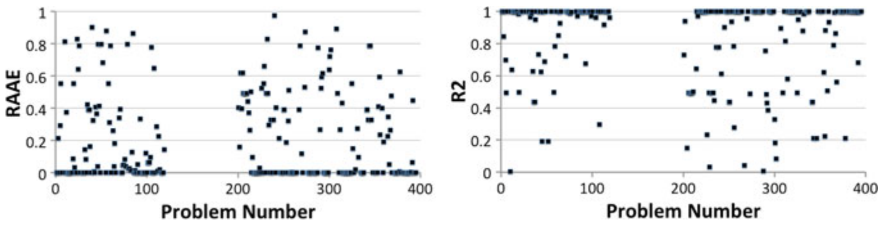

Fig. 2.11 Metrics for the FP-RBF method (scarce set of data)

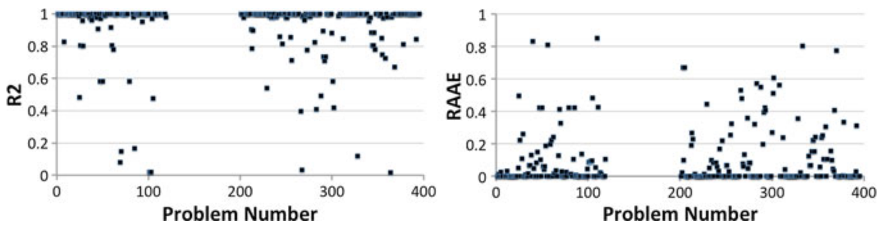

Fig. 2.12 Metrics for the FP-RBF method (*small* set of data)

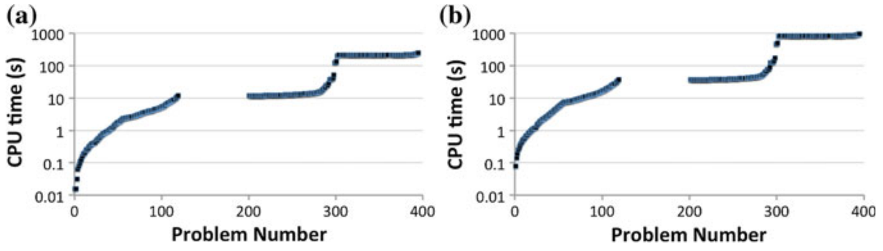

Fig. 2.13 CPU time for the FP-RBF method. **a** scarce set of data, **b** small set of data

have high values of R2 and low values of RAAE, indicating a good approximation, even for such small number of training points. When the number of training points is increased from scarce to small, results look better, as one can notice from the analysis of Fig. 2.12 for the R2 and RAAE metrics.

Figure 2.13 shows the CPU time for FP-RBF method. All test cases ran on an Intel i7 2GHz with 4 Gb of RAM (Windows XP emulated under Mac OS X 10.8.4) and codes were written in Fortran 90. Besides running on different processors, CPU times in these test cases are a little higher than in previous one, since now we are also optimizing the shape parameter, the RBF polynomial degree and the scaling of the variables, as discussed before. Notice that computing time increased only slightly by going from a scarce data set to a small data set.

Figure 2.14 shows difference of R2 and RAAE metrics between the Kriging method and the FP-RBF model for 296 test-cases studied in this section, using the scarce set of data. Since higher R2 and lower RAAE values indicate a good accuracy, in these graphics, negative values of Delta R2 and positive values of RAAE indicate

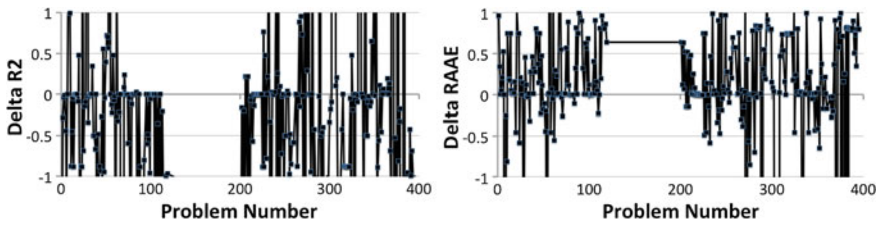

Fig. 2.14 Difference of the R2 and RAAE metrices between the Kriging model proposed by Jones et al. [23] and FP-RBF methods for the scarce set of data (negative values of Delta R2 and positive values of Delta RAAE indicates superiority of the FP-RBF method)

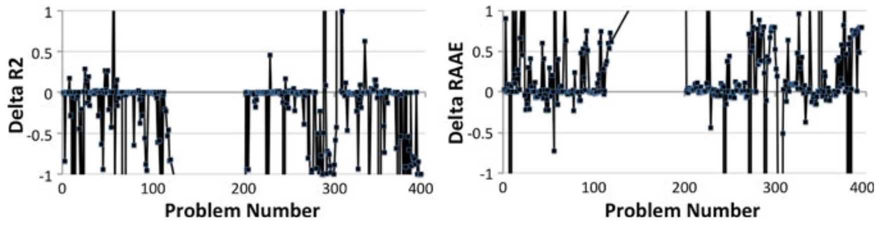

Fig. 2.15 Difference of the R2 and RAAE metrics between the Kriging model proposed by Jones et al. [23] and FP-RBF methods for the small set of data (negative values of Delta R2 and positive values of Delta RAAE indicates superiority of the FP-RBF method)

the superiority of the FP-RBF method. As a general trend, the FP-RBF method performs better than the original Kriging method, given by Eq. (2.9), for most of the test cases, although there are some functions were Kriging has a better accuracy. For the small set of data, such comparison is presented in Fig. 2.15, where one can notice the superiority of the FP-RBF method over Kriging.

Figure 2.16 shows the CPU time ratio between Kriging and FP-RBF for scarce and small sets of data. In this figure, values greater than one indicate how many times the Kriging is slower than the FP-RBF. As a general trend, for the scarce set of data, Kriging is one order of magnitude slower than the FP-RBF, whereas for the small set of data it is two orders of magnitude slower. Two factors contribute for the high computing cost of Kriging: (i) the need to invert the covariance matrix $\mathbf{R}$ in Eqs. (2.27)–(2.29) and the minimization of $[(\det\mathbf{R})^{1/m}\sigma^2]$ by the Particle Swarm method. We intend to investigate ways to reduce this computational cost, since Kriging seems to have some advantage over FP-RBF model when applied to certain functions, as shown above. It is also worth noting that the CPU time ratio is almost constant over all test problems. In fact, going from test function number 200–295 the CPU time ratio decreases, when the number of dimensions of the problems varies considerably, as one can check from Figs. 2.10 and 2.16.

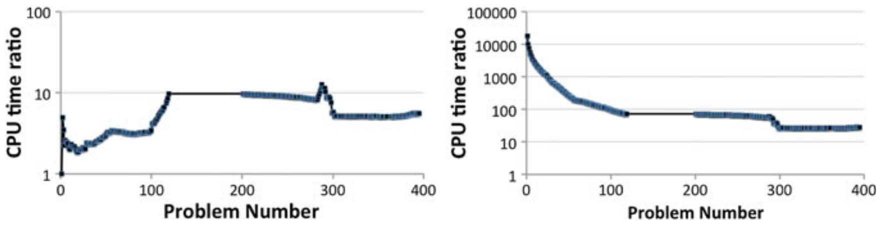

Fig. 2.16 CPU time ratio between the Kriging model proposed by Jones et al. [23] and FP-RBF methods for the scarce and small sets of data (values higher than one indicate the Kriging method is more expensive than FP-RBF)

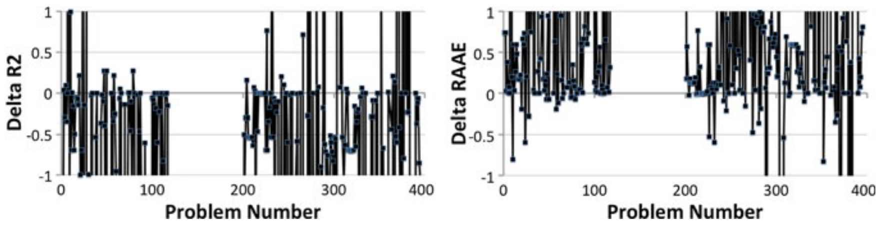

Fig. 2.17 Difference of the R2 and RAAE metrics between the HYBSORSM and FP-RBF methods for the scarce sets of data (negative values of Delta R2 and positive values of Delta RAAE indicates superiority of the FP-RBF method)

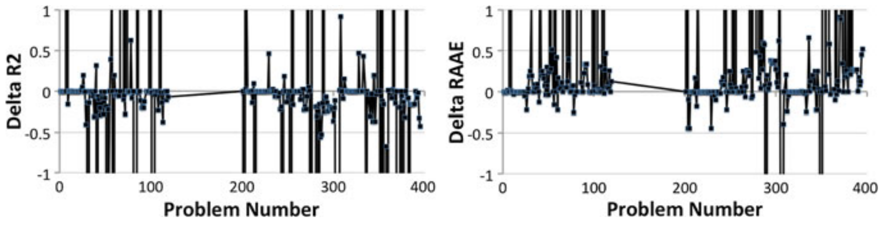

Fig. 2.18 Difference of the R2 and RAAE metrics between the HYBSORSM and FP-RBF methods for the small sets of data (negative values of Delta R2 and positive values of Delta RAAE indicates superiority of the FP-RBF method)

2.5.3 Fittest Polynomial RBF Versus Hybrid Self Organizing Response Surface Method—HYBSORSM

Results in Fig. 2.17 indicate better accuracy of the FP-RBF method over HYBSORM. From the analysis of this figure it is clear the superiority of FP-RBF model. Comparing Fig. 2.17 with Fig. 2.14, it is evident that for this set of data, the Kriging model is also superior to HYBSORSM. However, when the small set of data is used (see Fig. 2.18) the HYBSORM method improves its performance, but is still outperformed by the FP-RSM method. Comparing now Figs. 2.15 and 2.18, the HYBSORM has a better performance than the Kriging model for the small sets of data.

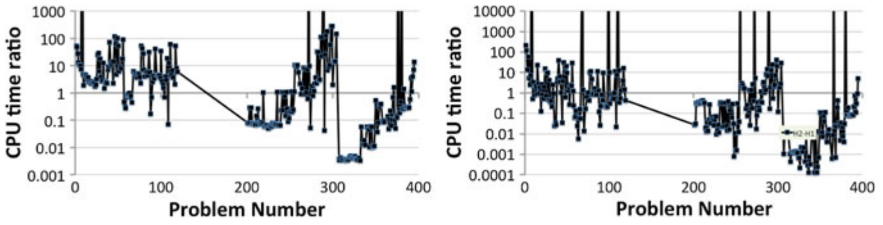

Fig. 2.19 CPU time ratio between the HYBSORSM and FP-RBF methods for the scarce and small sets of data (values higher than one indicate the method is more expensive than FP-RBF)

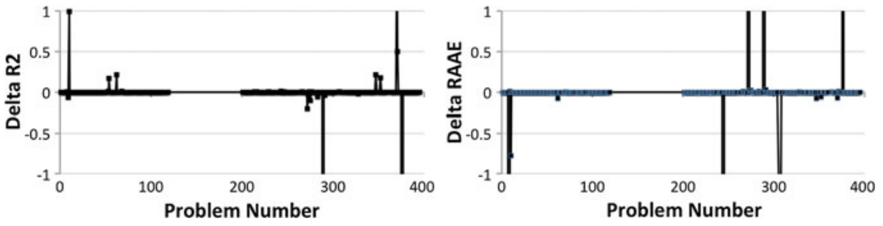

Fig. 2.20 Difference of the R2 and RAAE metrics between the KRG-FP-RBF and FP-RBF methods for the scarce sets of data (negative values of Delta R2 and positive values of Delta RAAE indicates superiority of the FP-RBF method)

Figure 2.19 shows some interesting results regarding the CPU time ratio between HYBSORM and FP-RSM model. In some cases, the first method is faster then the FP-RBF model, whereas it is slower for other ones. Comparing Figs. 2.10 and 2.19, it is interesting to notice that the CPU time ratio follows the same behaviour as the number of variables. Thus, opposite to the Kriging model, where the CPU time was almost constant with the number of variables, the HYBSORSM method requires more computational effort for problems where the number of dimensions is high.

2.5.4 *Fittest Polynomial RBF Versus Kriging Approximation with Fittest Polynomial Radial Basis Function—KRG-FP-RBF*

This section compares the results of the FP-RBF method with the ones obtained by the combined (hybrid) KRG-FP-RBF method. Figures 2.20 and 2.21 show Delta R2 and Delta RAAE for the scarce and small sets of data. In general, the hybrid KRG-FP-RBF method does not modify the accuracy of the FP-RBF method, except in a few cases. Although for some cases the performance decreases, for most of the cases where the hybrid KRG-FP-RBF method changes the FP-RBF performance, it improves the solution. These results are still in a very early stage of development and we believe this approach might improve the overall performance of the FP-RBF method if some better strategy to minimize $[(\det \mathbf{R})^{1/m}\sigma^2]$ is used.

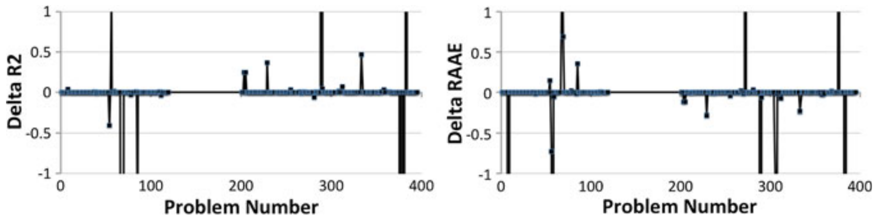

Fig. 2.21 Difference of the R2 and RAAE metrics between the KRG-FP-RBF and FP-RBF methods for the small set of data (negative values of Delta R2 and positive values of Delta RAAE indicates superiority of the FP-RBF method)

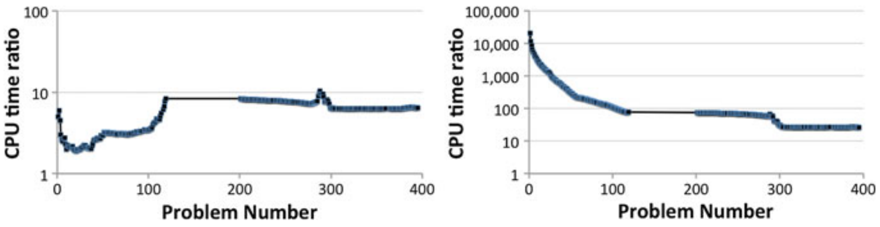

Fig. 2.22 CPU time ratio between the KRG-FP-RBF and FP-RBF methods for the scarce and small sets of data (values higher than one indicate the method is more expensive than FP-RBF)

Finally, Fig. 2.22 shows the CPU time ratio between the KRG-FP-RBF method and the FP-RBF method. This ratio can be reduced if better optimization procedures are used in the Kriging part of the code.

2.6 Conclusions

A number of concepts for constructing hybrid optimization algorithms with focus on automatic switching logic have been described. Also, a number of multi-dimensional response surface fitting algorithms and their hybrids have been described and their performances compared for scarce, small and medium data sets. Fittest polynomial radial basis function (FP-RBF) method appears to offer the best overall performance concerning high accuracy of fitting arbitrary data sets and low computing time requirements. Possible hybridization of Kriging and FP-RBF was also thoroughly tested showing its promises as far as increased robustness of such hybrids, although at significant increase in the computing time.

Acknowledgments This work was partially funded by the US Air Force Office of Scientific Research under grant FA9550-12-1-0440 monitored by Dr. Ali Sayir. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the US Air Force Office of Scientific Research or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes not with standing any copyright notation thereon. Dr. Colaço also acknowledges the financial support provided by CNPq, CAPES, FAPERJ and ANP/PRH37 (http://www.prh.mecanica.ufrj.br/), Brazilian agencies for fostering science and technology.

References

1. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. Evol Comput IEEE Trans Evol Comput 1(1):67–82
2. Dulikravich GS, Martin TJ, Colaço MJ, Inclan EJ (2013) Automatic switching algorithms in hybrid single-objective optimizers. FME Trans 41(3):167–179
3. Foster NF, Dulikravich GS (1997) Three-dimensional aerodynamic shape optimization using genetic and gradient search algorithms. AIAA J Spacecr Rockets 34(1):36–42
4. Dulikravich GS, Martin TJ, Dennis BH, Foster NF (1999) Multidisciplinary hybrid constrained GA optimization. In: Miettinen K, Makela MM, Neittaanmaki P, Periaux J (eds) EUROGEN99-evolutionary algorithms in engineering and computer science: recent advances and industrial applications, Jyvaskyla, Wiley, pp 233–259, 30 May–3 June 1999
5. Colaço MJ, Dulikravich GS, Orlande HRB, Martin TJ (2005) Hybrid optimization with automatic switching among optimization algorithms. In: Annicchiarico W, Périaux J, Cerrolaza M, Winter G (eds) Evolutionary algorithms and intelligent tools in engineering optimization, CIMNE, Barcelona, Spain/WITpress, pp 92–118
6. Ahrari A, Shariat-Panahi M, Atai AA (2009) GEM: a novel evolutionary optimization method with improved neighborhood search, Appl Math Comput 210(2):376–386
7. Deb K (2009) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester
8. Talbi E-G (2002) A taxonomy on hybrid metaheuristics. J Heuristics 8:541–562
9. Vrugt JA, Robinson BA (2007) Improved evolutionary optimization from genetically adaptive multimethod search. Proc Nat Acad Sci (US) 104(3):708–711
10. Moral RJ, Dulikravich GS (2008) Multi-objective hybrid evolutionary optimization with automatic switching among constituent algorithms. AIAA J 46(3):673–700
11. McKay M, Conover W, Beckman RA (1979) A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21:239–245
12. Sobol IM (1967) Distribution of points in a cube and approximate evaluation of integrals. USSR Comput Math Math Phys 7:86–112
13. Box GEP, Draper NR (1987) Empirical model-building and response surfaces,vol 157, Wiley Series in Probability and Statistics
14. Meyers RH, Montgomery DC, Anderson-Cook CM (2009) Response surface methodology: process and product optimization using designed experiments, vol 705, Wiley Series in Probability and Statistics
15. Matheron G (1963) Principles of geostatistics. Econ Geol 58:1246–1266
16. Hardy RL (1971) Multiquadric equations of topography and other irregular surfaces. J Geophys Res 176:1905–1915
17. Haykin S (2008) Neural networks and learning machines, Prentice Hall, Englewood Cliffs
18. Madalaand HR, Ivakhnenko AG (1994) Inductive learning algorithms for complex systems modeling. CRC Press, Boca Raton
19. Moral RJ, Dulikravich GS (2008) A hybrid self-organizing response surface methodology, paper AIAA-2008-5891. In: 12th AIAA/ISSMO multi-disciplinary analysis and optimization conference, Victoria, 10–12 September 2008
20. Krige DG (1951) A statistical approach to some basic mine valuation problems on the witwatersrand. J Chem Metal Mining Soc South Africa 52(6):119–139
21. Sacks J, Schiller SB, Welch WJ (1989) Design for computer experiments. Technometrics 31(1):41–47

22. Sacks J, Welch WJ, Mitchell TJ, Wynn HP (1989) Design and analysis of computer experiments. Stat Sci 4(4):409–435
23. Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. J Glob Optim 13:455–492
24. Huang D, Allen TT, Notz WI, Zeng N (2006) Global optimization of stochastic black-box systems via sequential kriging meta-models. J Glob Optim 34:441–466
25. Hardy RL (1971) Multiquadric equations of topography and other irregular surfaces. J Geophys Res 176:1905–1915
26. Wendland H (1998) Error estimates for interpolation by compactly supported radial basis functions of minimal degree. J Approx Theory 93:258–272
27. Colaço MJ, Dulikravich GS, Sahoo D (2008) A response surface method based hybrid optimizer. Inverse Prob Sci Eng 16:717–742
28. Buhmann MD (2003) Radial basis functions on grids and beyond. In: International workshop on meshfree methods, Lisbon, Portugal
29. Baxter BJC (1992) The interpolation theory of radial basis functions, PhD thesis, Cambridge
30. Zhang Q (1997) Using wavelet network in nonparametric estimation. IEEE Trans Neural Netw 8:227–236
31. Sahoo D, Dulikravich GS (2006) Evolutionary wavelet neural network for large scale function estimation in optimization, In: 11th AIAA/ISSMO multidisciplinary analysis and optimization conference, Portsmouth, 6–8 September 2006
32. Dulikravich GS, Egorov IN, Colaço MJ (2008) Optimizing chemistry of bulk metallic glasses for improved thermal stability. Model Simul Mater Sci Eng 16:075010
33. Bhargava S, Dulikravich GS, Murty GS, Agarwal A, Colaço MJ (2011) Stress corrosion cracking resistant aluminum alloys: optimizing concentrations of alloying elements and tempering. Mat Manuf Process 26:363–374
34. Colaço MJ, Dulikravich GS, Orlande HRB (2009) Magnetohydrodynamic simulations using radial basis functions. Int J Heat Mass Trans 52:5932–5939
35. Colaço MJ, Teixeira CV, Dutra LM (2010) Thermodynamic simulation and optimization of diesel engines operating with diesel and biodiesel blends using experimental data. Inverse Prob Sci Eng 18:787–812
36. Pires TS, Cruz MEC, Colaço MJ (2013) Response surface method applied to the thermoeconomic optimization of a complex cogeneration system modeled in a process simulator. Energy 52:44–54
37. Orlande HRB, Colaço MJ, Dulikravich GS (2008) Approximation of the likelihood function in the bayesian technique for the solution of inverse problems. Inverse Prob Sci Eng 16:677–692
38. IOSO NM (2003) Version 1.0, user's guide, IOSO Technology Center, Moscow, Russia
39. Colaço MJ, Silva WB, Magalhães AC, Dulikravich GS (2008) Response surface methods applied to scarce and small sets of training points A comparative study. In: EngOpt 2008 - international conference on engineering optimization, Rio de Janeiro, Brazil
40. Colaço MJ, Dulikravich GS (2008) A hybrid RBF based methods for highly multidimensional response surfaces using scarce data sets. In: 12th AIAA/ISSMO multidisciplinary analysis and optimization conference, Victoria, British Columbia
41. Jin R, Chen W, Simpson TW (2000) comparative studies of metamodeling techniques under multiple modeling criteria. In: 8th AIAA/USAF/NASA/ISSMO multidisciplinary analysis & optimization symposium, AIAA 2000–4801, Long Beach, 6–8 September 2000
42. Hock W, Schittkowski K (1981) Test examples for nonlinear programming codes. Lecture notes in economics and mathematical systems, vol 187. Springer, Berlin
43. Schittkowski K (1987) More test examples for nonlinear programming. Lecture notes in economics and mathematical systems, vol 282. Springer, Berlin
44. Colaço MJ, Dulikravich GS, Sahoo D (2007) A comparison of two methods for fitting high dimensional response surfaces. In: Inverse problems, design and optimization symposium, Miami, 16–18 April 2007