METTI IV – Thermal Measurements and Inverse Techniques
Rio de Janeiro, Brazil, November 8-13, 2009

A Survey of Basic Deterministic, Heuristic and Hybrid Methods for Single Objective Optimization and Response Surface Generation

Marcelo J. Colaço
Federal University of Rio de Janeiro
Dept. of Mechanical Engineering
Cid. Universitaria, Cx. Postal: 68503
Rio de Janeiro, RJ, 21941-972, BRAZIL
colaco@asme.org

George S. Dulikravich
Florida International University
Dept. of Mechanical and Materials Engineering
10555 West Flagler Street, EC 3462
Miami, FL 33174, USA
dulikrav@fiu.edu

Abstract

This paper surveys basic concepts for formulation and solution of single-objective optimization problems. Deterministic and stochastic minimization techniques in finite and infinite dimensional spaces are reviewed; advantages and disadvantages of each of them are discussed and some hybrid techniques are presented. A response surface methodology is also discussed, which can be used as a surrogate model in very complex high dimensionality problems.

1. Introduction

In this paper we address solution methodologies for single-objective optimization problems, based on minimization techniques. Several gradient-based and not gradient-based (stochastic) techniques are introduced, together with their basic implementation steps and algorithms. We present some deterministic methods, such as the Conjugate Gradient Method, the Newton Method and the Davidon-Fletcher-Powell Method [1-34]. In addition, we present some of the most promising stochastic approaches, such as the Simulated Annealing Method [35,36], the Differential Evolutionary Method [37], Genetic Algorithms [38,39] and the Particle Swarm Method [40-43]. Deterministic methods are in general computationally faster (they require fewer objective function evaluations in case of problems with low number of design variables) than stochastic methods, although they can converge to a local minima or maxima, instead of the global one. On the other hand, stochastic algorithms can ideally converge to a global maxima or minima, although they are computationally slower (for problems with relatively low number of design variables) than the deterministic ones. Indeed, the stochastic algorithms can require thousands of evaluations of the objective functions and, in some cases, become non-practical. In order to overcome these difficulties, we will also discuss the so-called hybrid algorithms [44-46], which take advantage of the robustness of the stochastic methods and the fast convergence of the deterministic methods [44-59]. Each technique provides a unique approach with varying degrees of convergence, reliability and robustness at different stages during the iterative minimization process. A set of analytically formulated rules and switching criteria can be coded into the program to automatically switch back and forth among the different algorithms as the iterative process advances [44-46].

In many optimization problems, evaluation of the objective function is extremely expensive and time consuming. For example, optimizing chemical concentrations of each of the alloying elements in a multi-component alloy requires manufacturing each candidate alloy and evaluating its properties using classical experimental techniques. Even with the most efficient optimization algorithms [47], this means that often hundreds of alloys having different chemical concentrations of their constitutive elements would have to be manufactured and tested. This is understandably too expensive to be economically acceptable. Similar is the situation when attempting to optimize three-dimensional aerodynamic shapes. Thousands of different aerodynamic shapes need to be analyzed using computational fluid dynamics software which would be unacceptably time consuming.

Therefore, for problems where objective function evaluations are already expensive and where the number of design variables is large thus requiring many such objective function evaluations, the only economically viable approach to optimization is to use an inexpensive and as accurate as possible surrogate model (a metamodel) instead of the actual high fidelity analysis method. Such surrogate models are known as response surfaces [46,60]. In case of more than three design variables, a response surface becomes a high dimensional hypersurface that needs to be fitted through the available (often small) set of high fidelity values of the objective function. Once the response surface (hypersurface) is created using an appropriate analytic formulation, it is very easy and fast to search such a surface for its minima given a set of values of design variables supporting such a response surface. Thus, we also present in this paper some basic concepts related to the response surface generation methodology.

2. Basic Concepts

2.1. Objective Function

The first step in establishing a procedure for the solution of either inverse problems or optimization problems is the definition of an *objective function*. The objective function is the mathematical representation of an aspect under evaluation, which must be minimized (or maximized). The objective function can be mathematically stated as

$$S = S(\mathbf{P}); \quad \mathbf{P} = \{P_1, P_2, ..., P_N\} \tag{1}$$

where P_1, P_2, … , P_N are the variables of the problem under consideration that can be modified in order to find the minimum value of the function S.

The relationship between S and **P** can, most of the times, be expressed by a physical / mathematical model. However, in some cases this relationship is impractical or even impossible and the variation of S with respect to **P** must be determined by experiments.

2.2. Unimodal versus Multimodal Objective Functions

Some of the methods that will be discussed here are only applicable to certain types of functions, namely unimodal, which are those having only one maximum (or a minimum) inside the range of parameters being analyzed. This does not mean that the function must be continuous, as one can see from the figure below, where the first two functions are unimodals. The third function is unimodal in the interval $0 < P < 3\pi/2$ and the forth function is multimodal.

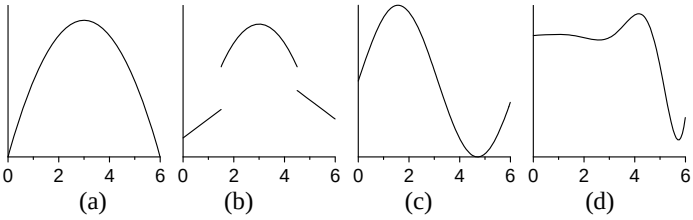


Figure 1 – Some examples of functions S (ordinate) of a single design variable P (abscissa)

For unimodal functions it is extremely easy to eliminate parts of the domain being analyzed in order to find the place of the maximum or minimum. Consider, for example, the first function of figure 1: if we are looking for the maximum value of the function, and we know that $S(P=1)$ is less than $S(P=2)$, we can immediately eliminate the region to the left of $P=1$, since the function is monotonically increasing its value. This is not true for multimodal functions, sketched as the forth function in figure 1.

2.3. Single and Multi-Objective Functions

This introduction will deal only with single-objective functions. However, it is interesting to introduce the reader to the mufti-objective optimization problems [39] since their applications in industry are very important. Consider, for example, the project of development of an automobile. Usually, we are not interested in only minimizing or maximizing a single function (fuel consumption, for example), but extremizing a large number of objective functions as, for example: fuel consumption, automobile weight, final price, performance, etc. This problem is called a multi-objective optimization and it is more complex than the case of a single-objective optimization.

In an aero-thermo-elasticity problem, for example, several disciplines are involved with various (often conflicting) objective functions to be optimized simultaneously. This case can be illustrated by the figure 2.

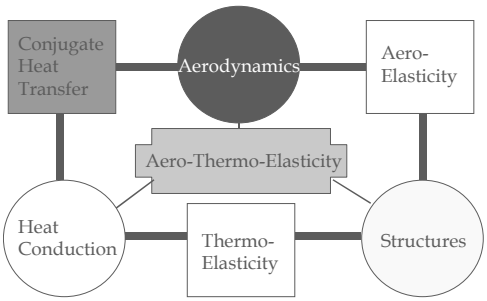


Figure 2 – An example of a multi-objective design optimization problem

2.4. Constraints

Usually the variables $P_1, P_2, \ldots, P_N$ which appear in the objective function formulation, are only allowed to vary within some pre-specified ranges. Such *constraints* are, for example, due to physical or economical limitations.

We can have two types of constraints. The first one is the *equality constraint*, which can be represented by

$$G = G(\mathbf{P}) = \varepsilon \tag{2}$$

This kind of constraint can represent, for example, the pre-specified power of an automobile.

The second type of constraint is called *inequality constraint* and it is represented by

$$G = G(\mathbf{P}) \leq \varepsilon \tag{3}$$

This can represent, for example, the maximum temperature allowed in a gas turbine engine.

2.5. Optimization Problems

For optimization problems, the objective function S can be, for example, the fuel consumption of an automobile and the variables $P_1, P_2, \ldots, P_N$ can be the aerodynamic profile of the car, the material of the engine, the type of wheels used, the distance from the floor, etc.

In this paper, we present deterministic and stochastic techniques for the minimization of an objective function $S(\mathbf{P})$ and the identification of the parameters $P_1, P_2, \ldots, P_N$, which appear in the objective function formulation. This type of minimization problem is solved in a space of finite dimension N, which is the number of unknown parameters. For many minimization problems, the unknowns cannot be recast in the form of a finite number of parameters and the minimization needs to be performed in an infinite dimensional space of functions [1-12,61-71].

3. Deterministic Methods

In this section some deterministic methods like the Steepest Descent method, the Conjugate Gradient method, the Newton-Raphson and quasi Newton methods will be discussed. Some practical aspects and limitations of such methods will be addressed.

These types of methods, as applied to non-linear minimization problems, generally rely on establishing an iterative procedure, which, after a certain number of iterations, will hopefully converge to the minimum of the objective function. The iterative procedure can be written in the following general form [2,3,5,17-20,32,33]:

$$\mathbf{P}^{k+1} = \mathbf{P}^k + \alpha^k \mathbf{d}^k \tag{4}$$

where $\mathbf{P}$ is the vector of design variables, α is the search step size, $\mathbf{d}$ is the direction of descent and k is the number of iterations.

An iteration step is *acceptable* if $S^{k+1} < S^k$. The direction of descent $\mathbf{d}$ will generate an acceptable step if and only if there exists a positive definite matrix $\mathbf{R}$, such that $\mathbf{d} = -\mathbf{R}\nabla S$ [32].

Such requirement results in directions of descent that form an angle greater than 90^o with the gradient direction. A minimization method in which the directions are obtained in this manner is called an *acceptable gradient method* [32].

A *stationary point* of the objective function is one at which $\nabla S = 0$. The most that we can hope for any gradient based method is that it converges to a stationary point. Convergence to the true minimum can be guaranteed only if it can be shown that the objective function has no other stationary points. In practice, however, one usually reaches the local minimum in the valley where the initial guess for the iterative procedure was located [32].

3.1. Steepest Descent Method [16-19]

The most basic gradient-based method is the Steepest Descent method. Some of the concepts developed here will be used in the next sections, where we will discuss more advanced methods. The basic idea of this method is to "walk" in the opposite direction of the locally highest variation of the objective function, in order to locate the minimum value of it. This can be exemplified in figure 3.

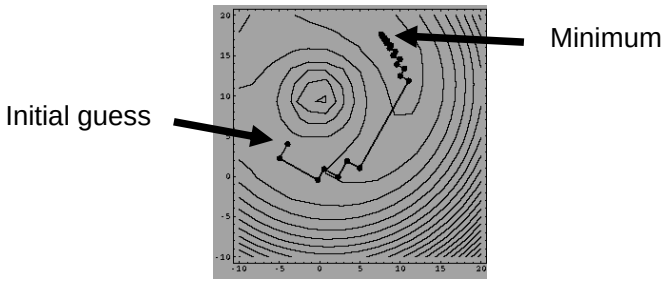


Figure 3 – Convergence history for the Steepest Descent method.

The objective function can be mathematically stated as

$$S = S(\mathbf{P}); \quad \mathbf{P} = \{P_1, P_2, ..., P_N\} \tag{5}$$

The direction in which the objective function S varies most rapidly is the direction of gradient of S. Thus, for the case with two variables (figure 3) the gradient is

$$\nabla S = \frac{\partial S}{\partial P_1}\mathbf{i}_1 + \frac{\partial S}{\partial P_2}\mathbf{i}_2 \tag{6}$$

The iterative process for finding the minimum value of the objective function can be written in the most general terms as

$$\mathbf{P}^{k+1} = \mathbf{P}^k + \alpha^k \mathbf{d}^{k+1} \tag{7}$$

where $\mathbf{P}$ is the vector of variables being optimized, α is the search step size, $\mathbf{d}$ is the direction of descent and k is a counter for the iterations. For the Steepest Descent method, the direction of descent is given by

$$\mathbf{d}^{k+1} = -\nabla S(\mathbf{P}^k) \tag{8}$$

In spite of this being the natural choice for the direction of descent, it is not very efficient as can be seen in Figure 1. Usually, the method starts with large variations in the objective function. As the minimum of the objective function is being approached, the convergence rate of this method becomes very low.

The optimum choice for the search step size is the one that causes the maximum variation in the objective function. Thus, using the iterative procedure given by equation (7) and the definition of the objective function (1), we have that at iteration level $k + 1$

$$S(\mathbf{P}^{k+1}) = S(\mathbf{P}^k + \alpha^k \mathbf{d}^{k+1}) \tag{9}$$

The optimum value of the step size α is obtained by solving

$$\frac{dS\left(\mathbf{P}^{k+1}\right)}{d\alpha^k} = 0 \tag{10}$$

Using the chain rule

$$\frac{dS\left(\mathbf{P}^{k+1}\right)}{d\alpha^k} = \frac{dS\left(P_1^{k+1}\right)}{dP_1^{k+1}}\frac{dP_1^{k+1}}{d\alpha^k} + \frac{dS\left(P_2^{k+1}\right)}{dP_2^{k+1}}\frac{dP_2^{k+1}}{d\alpha^k} + \cdots + \frac{dS\left(P_N^{k+1}\right)}{dP_N^{k+1}}\frac{dP_N^{k+1}}{d\alpha^k} \tag{11}$$

or

$$\frac{dS\left(\mathbf{P}^{k+1}\right)}{d\alpha^k} = \left\langle \left[\nabla S\left(\mathbf{P}^{k+1}\right)\right]^T, \frac{d\mathbf{P}^{k+1}}{d\alpha^k} \right\rangle \tag{12}$$

However, from equation (7) it follows that

(13)

$$\frac{d\mathbf{P}^{k+1}}{d\alpha^k} = \mathbf{d}^{k+1} = -\nabla S\left(\mathbf{P}^k\right)$$

Substituting equation (13) into (12) and (10), it follows that for Steepest Descent (figure 4)

$$\left\langle \left[\nabla S\left(\mathbf{P}^{k+1}\right)\right]^T, \nabla S\left(\mathbf{P}^k\right)\right\rangle = 0 \tag{14}$$

Thus, the optimum value of the search step size is the one that makes the gradients of the objective function at two successive iterations mutually orthogonal (figure 3).

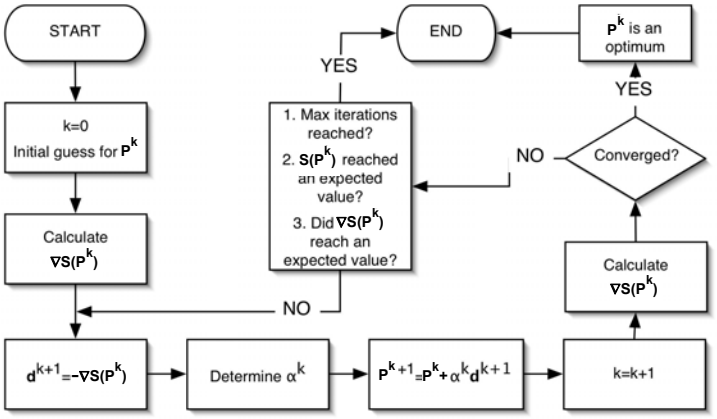


Figure 4 – Iterative procedure for the Steepest Descent method.

In "real life" applications, it is not possible to use equation (14) to evaluate the search step size, **d**. Thus, some univariate search methods need to be employed in order to find the best value of the search step size at each iteration. In the case of a unimodal function, some classical procedures can be used, such as the dichotomous search [17,18], Fibonacci search [17,18], golden search [17,18] and cubic spline interpolation [73], among others. However, for some realistic cases, the variation of the objective function with the search step size is not unimodal and then, more robust techniques are presented. The first one is the exhaustive search method and the second one is a technique based on exhaustive interpolation.

(a) Exhaustive Search [17,18]

This method is one of the less efficient search methods available for sequential computation (which means not parallel computation). However, it is the most utilized approach. Let us suppose, for example, that we are on a highway searching for a gas station with the lowest price of gasoline within an interval of five miles. If we do not have a newspaper or a telephone, the best way to do this is to go to each gas station and check the price and then determine the lowest value. This is the basis of the Exhaustive Search method. This method serves as an introduction to the next method, which is based on splines.

The basic idea consists in uniformly dividing the domain that we are interested in (the initial uncertainty region), and finding the region where the maximum or minimum value are located. Let us call this domain I_0. Let us suppose, for instance, the situation shown in figure 5, where an uncertainty interval I_0 was divided into eight sub regions, which are not necessarily the same size.

The objective function is evaluated at each of the nine points shown in the previous figure. From this analysis, we obtain

$$\begin{aligned} &y_1 < y_2 < y_3 < y_4 < y_5 \\ &y_5 > y_6 > y_7 > y_8 > y_9 \end{aligned} \tag{15}$$

Thus, the maximum point must be located between x_4 and x_6. Notice that we cannot say that the optimum is located between x_4 and x_5, nor between x_5 and x_6, since only a more refined grid could indicate this.

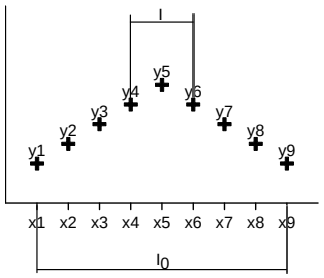


Figure 5 – Exhaustive Search method.

Thus, the final uncertainty interval I is (x_6-x_4) and the optimum point is located somewhere inside this interval. It can be shown [17,18] that I is given by

$$I = \frac{2I_0}{n+1} \tag{16}$$

where n is the number of objective functions evaluated. Notice that, once I is found, the process can be restarted making $I_0 = I$ and a more precise location for the maximum can be found. However, its precise location can never be reached.

In terms of sequential computation, this method is very inefficient. However, if we have a hypothetically large number of computers, all objective functions at each point in I_0 can be evaluated at the same time. Thus, for the example shown in figure 5, for n = 9, if we can assign the task of calculating the objective function at each point to an individual computer, the initial uncertainty region is reduced by 5 times within the time needed to just perform one calculation of the entire region using a single computer. Other more sophisticated methods, such as the Fibonacci method, for example, need sequential evaluations of the objective function. The Fibonacci method, for example, requires four objective function evaluations for the same reduction of the uncertainty region. Thus, in spite of its lack of efficiency in single processor applications, the Exhaustive Search method may be very efficient in parallel computing applications. A typical parallel computing arrangement is where one computer is the master and the other computers perform the evaluations of the objective function at each of the locations. A typical arrangement for the case depicted in figure 5 is presented in figure 6 where there are 10 computers; one of them being the master and the other nine performing the evaluations of the objective functions at the nine locations shown on figure 5.

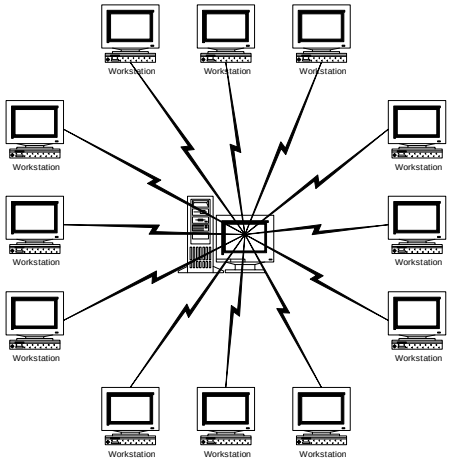


Figure 6 – Typical setup for a parallel computing

(b) Exhaustive Interpolation Search

This method is an improvement over the previous one, in that it requires fewer calculations to find the location of the minima. The method starts as the previous one, where domain is divided into several regions, where the objective functions are evaluated. The objective function is evaluated at a number of points in this domain. Next, a large number of points needs to be generated inside this domain and the objective function at these new points is estimated by spline fitting at the original points and interpolating at the new points using cubic splines [73], B-splines [74], kriging [75] or other interpolants. Interrogating these interpolated values we can find the region where the maximum or minimum values are located. The process can be repeated until a sufficiently small interval of uncertainty is obtained.

3.2. Conjugate Gradient Method [3,13,16-19,21-25,58,75-86]

The Steepest Descent method, in general, converges slowly for non-quadratic functions, since optimum search step sizes produce orthogonal gradients between two successive iterations. The Conjugate Gradient method tries to improve the convergence rate of the Steepest Descent method by choosing the directions of descent that reach the minimum value of the objective function faster. The iterative process for this method is given by the same equation used in the Steepest Descent method, equation (7). The difference is in the formulation for the direction of descent, which, for the Conjugate Gradient method, is given as a conjugation of the gradient and the direction of descent of the previous iteration

$$\mathbf{d}^{k+1} = -\nabla S\left(\mathbf{P}^k\right) + \gamma^k \mathbf{d}^{k-1} + \psi^k \mathbf{d}^q \tag{17}$$

where γ^k and ψ^k are conjugation coefficients. The superscript q in equation (17) denotes the iteration number where a restarting strategy is applied to the iterative procedure of the Conjugate Gradient method. Restarting strategies for the conjugate gradient method of parameter estimation were suggested by Powell [21] in order to improve its convergence rate. Different versions of the Conjugate Gradient method can be found in the literature depending on the form used for the computation of the direction of descent given by equation (17) [3,16,21-25,75-83]. In the Fletcher-Reeves version [22], the conjugation coefficients γ^k and ψ^k are obtained from the following expressions [3,16,21,22,75,78,80-83]

$$\gamma^k = \frac{\left\|\nabla S\left(\mathbf{P}^k\right)\right\|^2}{\left\|\nabla S\left(\mathbf{P}^{k-1}\right)\right\|^2}, \text{ with } \gamma^0 = 0 \text{ for } k=0 \tag{18.a}$$

$$\psi^k = 0, \text{ for } k = 0,1,2,\ldots \tag{18.b}$$

In the Polak-Ribiere version of the Conjugate Gradient method [3,16,21,24,77-79] the conjugation coefficients are given by

$$\gamma^k = \frac{[\nabla S\left(\mathbf{P}^k\right)]^T \left[\nabla S\left(\mathbf{P}^k\right) - \nabla S\left(\mathbf{P}^{k-1}\right)\right]}{\left\|\nabla S\left(\mathbf{P}^{k-1}\right)\right\|^2}, \text{ with } \gamma^0 = 0 \text{ for } k=0 \tag{19.a}$$

$$\psi^k = 0, \text{ for } k = 0,1,2,\ldots \tag{19.b}$$

Based on a previous work by Beale [25], Powell [21] suggested the following expressions for the conjugation coefficients, which gives the so-called Powell-Beale's version of the Conjugate Gradient method [21,25,76]

$$\gamma^k = \frac{[\nabla S(\mathbf{P}^k)]^T [\nabla S(\mathbf{P}^k) - \nabla S(\mathbf{P}^{k-1})]}{[\mathbf{d}^{k-1}]^T [\nabla S(\mathbf{P}^k) - \nabla S(\mathbf{P}^{k-1})]}, \text{ with } \gamma^0 = 0 \text{ for } k=0 \tag{20.a}$$

$$\psi^k = \frac{[\nabla S(\mathbf{P}^k)]^T [\nabla S(\mathbf{P}^{q+1}) - \nabla S(\mathbf{P}^q)]}{[\mathbf{d}^q]^T [\nabla S(\mathbf{P}^{q+1}) - \nabla S(\mathbf{P}^q)]}, \text{ with } \gamma^0 = 0 \text{ for } k=0 \tag{20.b}$$

In accordance with Powell [21], the application of the conjugate gradient method with the conjugation coefficients given by equations (20) requires restarting when gradients at successive iterations tend to be non-orthogonal (which is a measure of the local non-linearity of the problem) and when the direction of descent is not sufficiently downhill. Restarting is performed by making $\psi^k = 0$ in equation (17).

The non-orthogonality of gradients at successive iterations is tested by the following equation

$$ABS\left([\nabla S(\mathbf{P}^{k-1})]^T \nabla S(\mathbf{P}^k)\right) \geq 0.2 \left\|\nabla S\left(\mathbf{P}^k\right)\right\|^2 \tag{21.a}$$

where ABS (.) denotes the absolute value.

A non-sufficiently downhill direction of descent (i.e., the angle between the direction of descent and the negative gradient direction is too large) is identified if either of the following inequalities is satisfied

$$[\mathbf{d}^k]^T \nabla S\left(\mathbf{P}^k\right) \leq -1.2 \left\|\nabla S\left(\mathbf{P}^k\right)\right\|^2 \tag{21.b}$$

$$[\mathbf{d}^k]^T \nabla S\left(\mathbf{P}^k\right) \geq -0.8 \left\|\nabla S\left(\mathbf{P}^k\right)\right\|^2 \tag{21.c}$$

We note that the coefficients 0.2, 1.2 and 0.8 appearing in equations (21.a-c) are empirically determined and are the

same values used by Powell [21].

In Powell-Beale's version of the conjugate gradient method, the direction of descent given by equation (17) is computed in accordance with the following algorithm for $k \geq 1$ [21]:

STEP 1: Test the inequality (21.a). If it is true, set $q = k$-1.
STEP 2: Compute γ^k using equation (20.a).
STEP 3: If $k = q$+1, set $\psi^k = 0$. If $k \neq$ q+1 compute ψ^k using equation (20.b).
STEP 4: Compute the search direction $\mathbf{d}^{k+1}$ using equation (17).
STEP 5: If $k \neq q$+1, test the inequalities (21.b,c). If either one of them is satisfied, set $q = k$-1 and ψ^k=0. Then, recompute the search direction using equation (17).

The Steepest Descent method, with the direction of descent given by the negative gradient equation, would be recovered with $\gamma^k = \psi^k = 0$ for any k in equation (17). We note that the conjugation coefficients γ^k given by equations (18.a), (19.a) and (20.a) are equivalent for quadratic functions, because the gradients at different iterations are mutually orthogonal [16,21].

The same procedures used for the evaluation of the search step size in the Steepest Descent method can be employed here. Figure 7 illustrates the convergence history for the Fletcher-Reeves version of the Conjugate Gradient method for the same function presented in figure 3. One can see that the Conjugate Gradient method is faster than the Steepest Descent. It is worth noting that the gradients between two successive iterations are no longer mutually orthogonal.

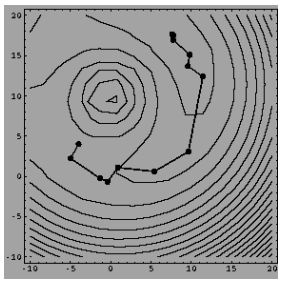


Figure 7 – Convergence history for the Fletcher-Reeves version of the Conjugate Gradient method.

Colaço and Orlande [15] presented a comparison of Fletcher-Reeves', Polak-Ribiere's and Powell-Beale's versions of the conjugate gradient method, as applied to the estimation of the heat transfer coefficient at the surface of a plate. This inverse problem was solved as a function estimation approach, by assuming that no information was available regarding the functional form of the unknown. Among the three versions tested for the Conjugate Gradient method, the method suggested by Powell and Beale appeared to be the best, as applied to the cases examined in that paper. This algorithm did not present the anomalous increase of the functional as observed with the other versions, and its average rates of reduction of the functional were the largest. As a result, generally, the smallest values for the RMS error of the estimated functions were obtained with Powell-Beale's version of the conjugate gradient method.

Figure 8 shows the iterative procedure for the Fletcher-Reeves version [22] of the Conjugate Gradient method.

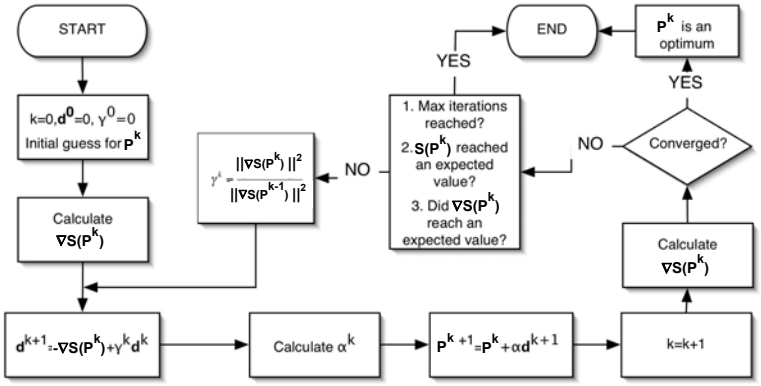


Figure 8 – Iterative procedure for the Fletcher-Reeves version of the Conjugate Gradient method.

3.3. Newton-Raphson Method [16-19]

While the Steepest Descent and the Conjugate Gradient methods use gradients of the objective function in their iterative procedures, the Newton-Raphson method uses information of the second derivative of the objective function in order to achieve a faster convergence rate (which does not necessarily mean a shorter computing time).

Let us consider a function $S(\mathbf{P})$, which is at least twice differentiable. The Taylor expansion of $S(\mathbf{P})$ around a vector $\mathbf{h}$ is given by

$$S(\mathbf{P}+\mathbf{h}) = S(\mathbf{P}) + \nabla S(\mathbf{P})^T \mathbf{h} + \frac{1}{2}\mathbf{h}^T \mathrm{D}^2 S(\mathbf{P})\mathbf{h} + O(\mathbf{h}^3) \tag{22}$$

where $\nabla S(\mathbf{P})$ is the gradient (vector of 1st order derivatives) while $\mathrm{D}^2 S(\mathbf{P})$ is the Hessian (matrix of 2nd order derivatives).

If the objective function $S(\mathbf{P})$ is twice differentiable, then the Hessian is always symmetrical, and we can write

$$\nabla S(\mathbf{P}+\mathbf{h}) \cong \nabla S(\mathbf{P}) + \mathrm{D}^2 S(\mathbf{P})\mathbf{h} \tag{23}$$

The optimum is obtained when the left side of equation (23) vanishes. Thus, we have

$$\mathbf{h}_{optimum} \cong -\left[\mathrm{D}^2 S(\mathbf{P})\right]^{-1} \nabla S(\mathbf{P}) \tag{24}$$

and the vector that optimizes the function $S(\mathbf{P})$ is

$$\left(\mathbf{P}+\mathbf{h}_{optimum}\right) \cong \mathbf{P} - \left[\mathrm{D}^2 S(\mathbf{P})\right]^{-1} \nabla S(\mathbf{P}) \tag{25}$$

Thus, introducing a search step size, which can be used to control the rate of convergence of the method, we can rewrite the Newton-Raphson method in the form of the equation (7) where the direction of descent is given by

$$\mathbf{d}^{k+1} = -\left[\mathrm{D}^2 S(\mathbf{P}^k)\right]^{-1} \nabla S(\mathbf{P}^k) \tag{26}$$

The Newton-Raphson method is faster than the Conjugate Gradient method as demonstrated in figure 9. However, the calculation of the Hessian matrix coefficients takes a long time. Figure 10 shows the iterative procedure for the Newton-Raphson method. Some other methods which do not require second order derivatives, so-called quasi Newton methods, will be addressed in the next section.

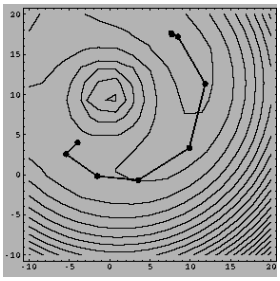


Figure 9 – Convergence history for the Newton-Raphson method.

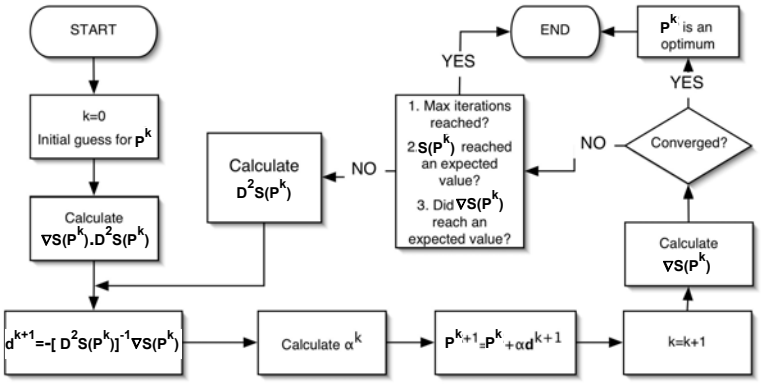


Figure 10 – Iterative procedure for the basic Newton-Raphson method implementation.

3.4. Quasi Newton Methods [16-19]

These kinds of methods try to calculate the Hessian appearing in the Newton-Raphson method in a manner that does not involve second order derivatives. Usually, they employ approximation for the Hessian based only on first order derivatives. Thus, they have a slower convergence rate than the Newton-Raphson method, but they are overall computationally faster.

Let us define a new matrix **H**, which is an approximation to the inverse of the Hessian as

$$\mathbf{H}^k = \left[\mathrm{D}^2 S(\mathbf{P}^k)\right]^{-1} \tag{27}$$

Thus, the quasi Newton methods follow the general iterative procedure given by equation (7) where the direction of descent is given by

$$\mathbf{d}^{k+1} = -\mathbf{H}^k \nabla S\left(\mathbf{P}^k\right) \tag{28}$$

The matrix **H** for quasi-Newton methods is iteratively calculated as

$$\mathbf{H}^k = \mathbf{H}^{k-1} + \mathbf{M}^{k-1} + \mathbf{N}^{k-1} \text{ for } k = 1,2,\ldots \tag{29.a}$$

$$\mathbf{H}^k = \mathbf{I} \text{ for } k = 0 \tag{29.b}$$

where **I** is the identity matrix. This means that during the first iteration, the quasi-Newton method starts as the Steepest Descent method.

Different quasi-Newton methods can be found depending on the choice for the matrices **M** and **N**. For the Davidon-Fletcher-Powell (DFP) method [26,27], such matrices are given by

$$\mathbf{M}^{k-1} = \alpha^{k-1} \frac{\mathbf{d}^{k-1}\left(\mathbf{d}^{k-1}\right)^T}{\left(\mathbf{d}^{k-1}\right)^T \mathbf{Y}^{k-1}} \tag{30.a}$$

$$\mathbf{N}^{k-1} = -\frac{\left(\mathbf{H}^{k-1}\mathbf{Y}^{k-1}\right)\left(\mathbf{H}^{k-1}\mathbf{Y}^{k-1}\right)^T}{\left(\mathbf{Y}^{k-1}\right)^T \mathbf{H}^{k-1}\mathbf{Y}^{k-1}} \tag{30.b}$$

where

$$\mathbf{Y}^{k-1} = \nabla S\left(\mathbf{P}^k\right) - \nabla S\left(\mathbf{P}^{k-1}\right) \tag{30.c}$$

Figure 11 shows the results for the minimization of the objective function shown before, using the DFP method. One can see that its convergence rate is between the Conjugate Gradient method and the Newton-Raphson method.

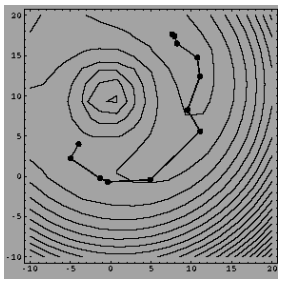


Figure 11 – Convergence history for the DFP method.

Note that, since the matrix **H** is iteratively calculated, some errors can be propagated and, in general, the method needs to be restarted after certain number of iterations [39]. Also, since the matrix **M** depends on the choice of the search step size α, the method is very sensitive to its value.

A variation of the DFP method is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [26-29], which is less sensitive to the choice of the search step size. For this method, the matrices **M** and **N** are calculated as

$$\mathbf{M}^{k-1} = \left(\frac{1 + \left(\mathbf{Y}^{k-1}\right)^T \mathbf{H}^{k-1} \mathbf{Y}^{k-1}}{\left(\mathbf{Y}^{k-1}\right)^T \mathbf{d}^{k-1}} \right) \frac{\mathbf{d}^{k-1} \left(\mathbf{d}^{k-1}\right)^T}{\left(\mathbf{d}^{k-1}\right)^T \mathbf{Y}^{k-1}} \tag{31.a}$$

$$\mathbf{N}^{k-1} = -\frac{\mathbf{d}^{k-1} \left(\mathbf{Y}^{k-1}\right)^T \mathbf{H}^{k-1} + \mathbf{H}^{k-1} \mathbf{Y}^{k-1} \left(\mathbf{d}^{k-1}\right)^T}{\left(\mathbf{Y}^{k-1}\right)^T \mathbf{d}^{k-1}} \tag{31.b}$$

Figure 12 shows the iterative procedure for the BFGS method.

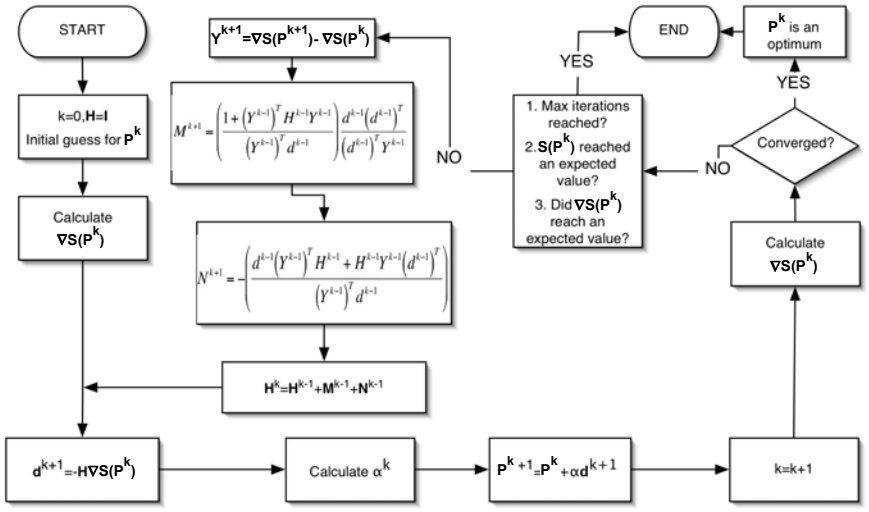
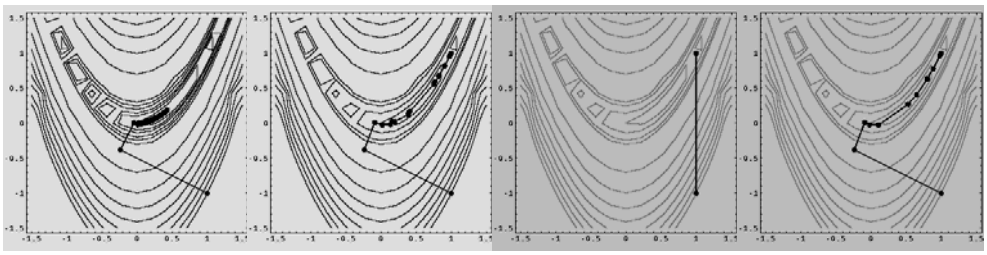


Figure 12 – Iterative procedure for the BFGS method.

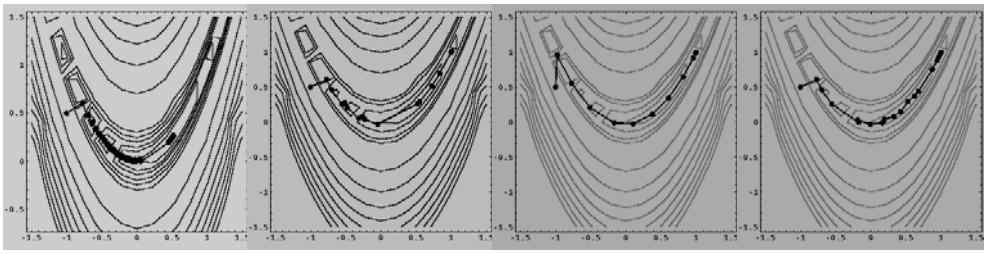
At this point it is of interest to explore the influence on the initial guess for the four methods introduced thus far. Usually, all these methods quickly converge to the minimum value if it is close to the initial guess. The Newton-Raphson method, however, without the search step size, moves to the extreme point closest to the initial guess, irregardless if it is a maximum, minimum or a saddle point. This is the reason why we introduce a search step size in equation (25). The search step size prevents the method from jumping to a maximum value when we look for a minimum and vice-versa. Figures 13 and 14 show the influence of the initial guess for all four methods for a Rosenbrock "banana-shape" function [104].

It should be pointed out that in real life situations, topology of the objective function space is not smooth and second derivatives of the objective function cannot be evaluated with any degree of confidence. Thus, all gradient-based and second derivative based search optimization algorithms have serious issue with robustness and reliability of their applications to realistic problems.



(a) (b) (c) (d)

Figure 13 – First initial guess for the (a) Steepest Descent, (b) Conjugate Gradient, (c) Newton-Raphson and (d) DFP methods.



(a) (b) (c) (d)

Figure 14 – Second initial guess for the (a) Steepest Descent, (b) Conjugate Gradient, (c) Newton-Raphson and (d) DFP methods.

3.5. Levenberg-Marquardt Method [30,31]

The Levenberg-Marquardt Method was first derived by Levenberg [30] in 1944, by modifying the ordinary least squares norm. Later, in 1963, Marquardt [31] derived basically the same technique by using a different approach. Marquardt's intention was to obtain a method that would tend to the Gauss method in the neighborhood of the minimum of the ordinary least squares norm, and would tend to the steepest descent method in the neighborhood of the initial guess used for the iterative procedure. This method actually converts a matrix that approximates the Hessian into a positive definite one, so that the direction of descent is acceptable.

The method rests on the observation that if **J** is a positive definite matrix, then $\mathbf{A} + \lambda \mathbf{J}$ is positive definite for sufficiently large λ. If **A** is an approximation for the Hessian, we can choose **J** as a diagonal matrix whose elements coincide with the absolute values of the diagonal elements of **A** [32].

The direction of descent for the Levenberg-Marquardt method is given by [32]

$$\mathbf{d}^k = -(\mathbf{A}^k + \lambda^k \mathbf{J}^k)^{-1} \nabla S(\mathbf{P}^k) \tag{32}$$

and the step size is taken as $\alpha^k = 1$. Note that for large values of λ^k a small step is taken along the negative gradient direction. On the other hand, as λ^k tends to zero, the Levenberg-Marquardt method tends to an approximation of Newton's method based on the matrix **A**. Usually, the matrix **A** is taken as that for the Gauss method [2,7,32].

4. Evolutionary and Stochastic Methods

In this section, some evolutionary and stochastic methods like Genetic Algorithm, Differential Evolution, Particle Swarm and Simulated Annealing will be discussed. Evolutionary methods, in contrast to the deterministic methods, do not rely, in general, on strong mathematical basis and do not make use of the gradient nor second derivative of the objective function as a direction of descent. The evolutionary optimization algorithms attempt to mimic nature in order to find the minimum of the objective function.

4.1. Genetic Algorithms [38]

Genetic algorithms are heuristic global optimization methods that are based on the process of natural selection. Starting from a randomly generated population of candidate designs, the optimizer seeks to produce improved designs from one generation to the next. This is accomplished by exchanging genetic information between designs in the current population, in what is referred to as the crossover operation. Hopefully this crossover produces improved designs, which are then used to populate the next generation [38,39].

The basic Genetic Algorithm works with a collection or population of candidate solutions to the optimization

problem. The algorithm works in an iterative manner. At each iteration, also called generation, three operators are applied to the entire population of designs. These operators are selection, crossover, and mutation. For the operators to be effective, each candidate solution or design must be represented as a collection of finite parameters, also called genes. Each design must have a unique sequence of these parameters that define it. This collection of genes is often called the chromosome. The genes themselves are often encoded as binary strings, though they can be represented as real numbers. The length of the binary string determines how precisely the value, also known as the allele, of the gene is represented.

The Genetic Algorithm applied to an optimization problem proceeds as follows. The process begins with an initial population of random designs. Each gene is generated by randomly generating 0's and 1's. The chromosome strings are then formed by combining the genes together. This chromosome string defines the design. The objective function is evaluated for each design in the population. Each design is assigned a fitness value, which corresponds to the value of the objective function for that design. In the minimization case, a higher fitness is assigned to designs with lower values of the objective function.

Next, the population members are selected for reproduction, based upon their fitness. The selection operator is applied to each member of the population. The selection operator chooses pairs of individuals from population who will mate and produce an offspring. In the tournament selection scheme, random pairs are selected from the population and the individual with the higher fitness of each pair is allowed to mate.

Once a mating pair is selected, the crossover operator is applied. The crossover operator essentially produces new designs or offspring by combining the genes from the parent designs in a stochastic manner. In the uniform crossover scheme, it is possible to obtain any combination of the two parent's chromosomes. Each bit in each gene in the chromosome is assigned a probability that crossover will occur (for example, 50 % for all genes). A random number between 0 and 1 is generated for each bit in each gene. If a number greater than 0.5 is generated, then that bit is replaced by the corresponding bit in the gene from the other parent. If it is less than 0.5, the original bit in the gene remains unchanged. This process is repeated for the entire chromosome for each of the parents. When complete, two offsprings are generated, which may replace the parents in the population.

The mutation process follows next. When the crossover procedure is complete and a new population is formed, the mutation operator is applied. Each bit in each gene in the design is subjected to a chance for a change from 0 to 1, or vice versa. The chance is known as the mutation probability, which is usually small. This introduces additional randomness into the process, which helps to avoid local minima. Completion of the mutation process signals the end of a design cycle. Many cycles may be needed before the method converges to an optimum design.

For more details or for the numerical implementation of Genetic Algorithms, the reader is referred to [38,39].

4.2. Differential Evolution [37]

The Differential Evolution method is an evolutionary method based on Darwin's theory of evolution of the species [87]. This non-gradient based optimization method was created in 1995 [37] as an alternative to the Genetic Algorithm methods. Following Darwin's theory, the strongest members of a population will be more capable of surviving in a certain environmental condition. During the mating process, the chromosomes of two individuals of the population are combined in a process called crossover. During this process mutations can occur, which can be good (individual with a better objective function) or bad (individual with a worse objective function). The mutations are used as a way to escape from local minima. However, their excessive usage can lead to a non-convergence of the method.

The method starts with a randomly generated population in the domain of interest. Thus, successive combinations of chromosomes and mutations are performed, creating new generations until an optimum value is found.

The iterative process is given by (figure 15)

$$\mathbf{P}_i^{k+1} = \delta_1 \mathbf{P}_i^k + \delta_2 \left[\boldsymbol{\alpha} + F(\boldsymbol{\beta} - \boldsymbol{\gamma}) \right] \tag{33}$$

where

$\mathbf{P}_i$ is the *i*-th individual of the vector of parameters.

$\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are three members of population matrix $\boldsymbol{\Pi}$, randomly choosen.

F is a weight function which defines the mutation ($0.5 < F < 1$).

k is a counter for the generations.

δ_1 and δ_2 delta Dirac functions that define the mutation.

In this minimization process, if $S(\mathbf{P}^{k+1}) < S(\mathbf{P}^k)$, then $\mathbf{P}^{k+1}$ replaces $\mathbf{P}^k$ in the population matrix $\boldsymbol{\Pi}$. Otherwise, $\mathbf{P}^k$ is kept in the population matrix.

The binomial crossover is given as

$$\begin{array}{ll} \delta_1 = 0, \text{ if } R < CR & \qquad \delta_2 = 1, \text{ if } R < CR \\ \quad 1, \text{ if } R > CR & \qquad \quad 0, \text{ if } R > CR \end{array} \tag{34.a,b}$$

where CR is a factor that defines the crossover (0.5 < CR < 1) and R is a random number with uniform distribution between 0 and 1.

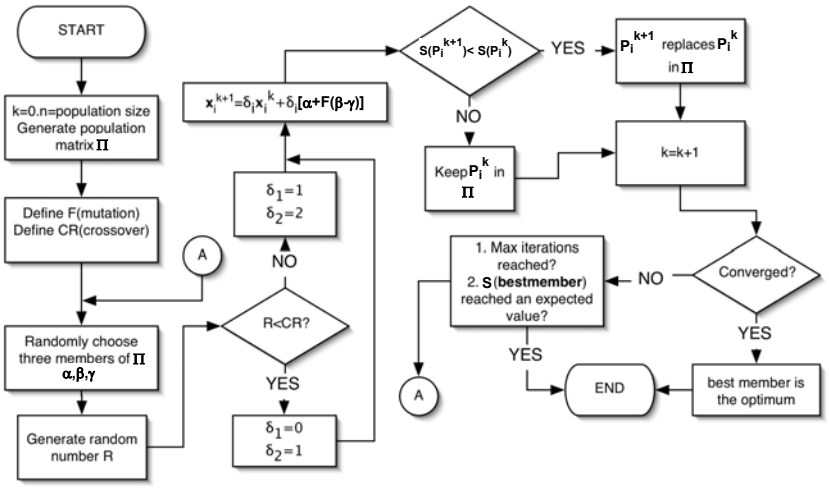


Figure 15 – Iterative procedure for the Differential Evolution method.

4.3. Particle Swarm [40-43]

This non-gradient based optimization method was created in 1995 by an electrical engineer (Russel Eberhart) and a social psychologist (James Kennedy) [40-43] as an alternative to the Genetic Algorithm methods. This method is based on the social behavior of various species and tries to equilibrate the individuality and sociability of the individuals in order to locate the optimum of interest. The original idea of Kennedy and Eberhart came from the observation of birds looking for a nesting place. When the individuality is increased, the search for alternative places for nesting is also increased. However, if the individuality becomes too high, the individual might never find the best place. In other words, when the sociability is increased, the individual learns more from their neighbor's experience. However, if the sociability becomes too high, all the individuals might converge to the first place found (possibly a local minima).

In this method, the iterative procedure is given by

$$\mathbf{P}_i^{k+1} = \mathbf{P}_i^k + \mathbf{v}_i^{k+1} \tag{35.a}$$

$$\mathbf{v}_i^{k+1} = \alpha \mathbf{v}_i^k + \beta\, \mathbf{r}_{1i}\left(\boldsymbol{\pi}_i - \mathbf{P}_i^k\right) + \beta\, \mathbf{r}_{2i}\left(\boldsymbol{\pi}_g - \mathbf{P}_i^k\right) \tag{35.b}$$

where:

$\mathbf{P}_i$ is the *i*-th individual of the vector of parameters.

$\mathbf{v}_i = 0$, for k = 0.

$\mathbf{r}_{1i}$ and $\mathbf{r}_{2i}$ are random numbers with uniform distribution between 0 and 1.

$\boldsymbol{\pi}_i$ is the best value found by the i-th individual, $\mathbf{P}_i$.

$\boldsymbol{\pi}_g$ is the best value found by the entire population.

$0 < \alpha < 1$; $1 < \beta < 2$

In equation (35.b), the second term on the right hand side represents the individuality and the third term the sociability. The first term on the right-hand side represents the inertia of the particles and, in general, must be decreased as the iterative process proceeds. In this equation, the vector $\boldsymbol{\pi}_i$ represents the best value ever found for the i-th component vector of parameters $\mathbf{P}_i$ during the iterative process. Thus, the individuality term involves the comparison between the current value of the i-th individual $\mathbf{P}_i$ and its best value in the past. The vector $\boldsymbol{\pi}_g$ is the best value ever found for the entire population of parameters (not only the i-th individual). Thus, the sociability term compares $\mathbf{P}_i$ with the best value of the entire population in the past.

Figure 16 shows the iterative procedure for the Particle Swarm method.

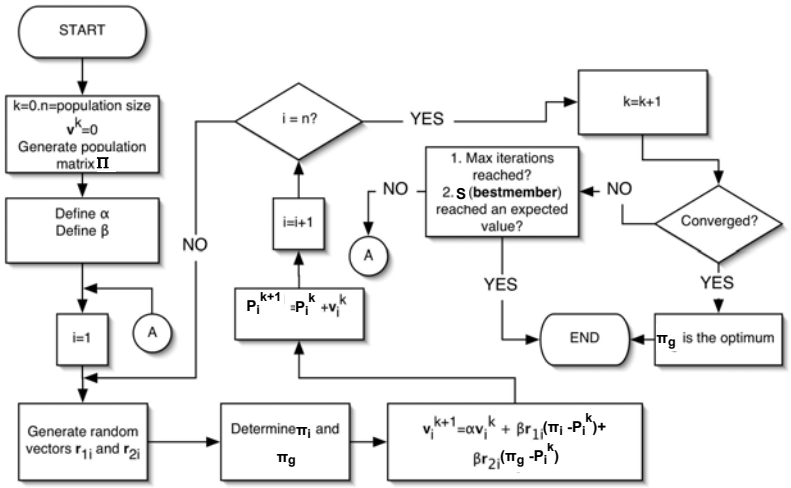


Figure 16 – Iterative procedure for the Particle Swarm method.

4.4. Simulated Annealing [35,36]

This method is based on the thermodynamics of the cooling of a material from a liquid to a solid phase. If a liquid material (e.g., liquid metal) is slowly cooled and left for a sufficiently long time close to the phase change temperature, a perfect crystal will be created, which has the lowest internal energy state.

On the other hand, if the liquid material is not left for a sufficient long time close to the phase change temperature, or, if the cooling process is not sufficiently slow, the final crystal will have several defects and a high internal energy state. This phenomena is similar to the quenching process used in metallurgical applications.

The gradient-based methods move in directions that successively lower the objective function value when minimizing the value of a certain function or in directions that successively raise the objective function value in the process of finding the maximum value of a certain function. The Simulated Annealing method can move in any direction at any point in the optimization process, thus escaping from possible local minimum or local maximum values.

We can say that gradient-based methods "cool down too fast", going rapidly to an optimum location which, in most cases, is not the global, but a local one. As opposed to gradient-based methods, nature works in a different way. Consider, for example, the Boltzmann probability function given as

$$\text{Prob}(E) \quad \propto \quad e^{(-E/KT)} \tag{36}$$

This equation expresses the idea that a system in thermal equilibrium has its energy distributed probabilistically among different energy states E where K is the Boltzmann constant. Equation (36) tells us that even at low temperatures, there is a chance, although small, that the system is at a high energy level, as illustrated in figure 17. Thus, there is a chance that the system could get out of this local minimum and continue looking for another one, possibly the global minimum.

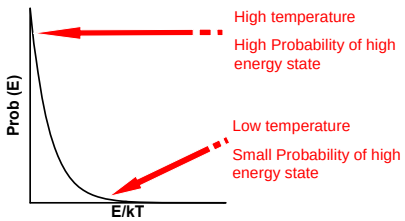


Figure 17 – Schematic representation of eq. (36)

Figure 18 shows the iterative procedure for the Simulated Annealing method. The procedure starts generating a population of individuals of the same size of the number of variables ($n = m$), in such a way that the population matrix is a square matrix. Then, the initial temperature (T), the reducing ratio (RT), the number of cycles (N_s) and the number of iterations of the annealing process (N_{it}) are selected. After N_s*n function evaluations, each element of the step length V is adjusted so that approximately half of all function evaluations are accepted. The suggested value for the number of cycles is 20. After N_{it}*N_s*n function evaluations, the temperature (T) is changed by the factor RT. The value suggested for the number of iterations by Corana *et al.* [35] is MAX(100, 5*n).

The iterative process follows the equation

$$P_i^1 = P_i^0 + RV_i \tag{37}$$

Here, R is a random number with a uniform distribution between 0 and 1 and V is a step-size which is continuously adjusted.

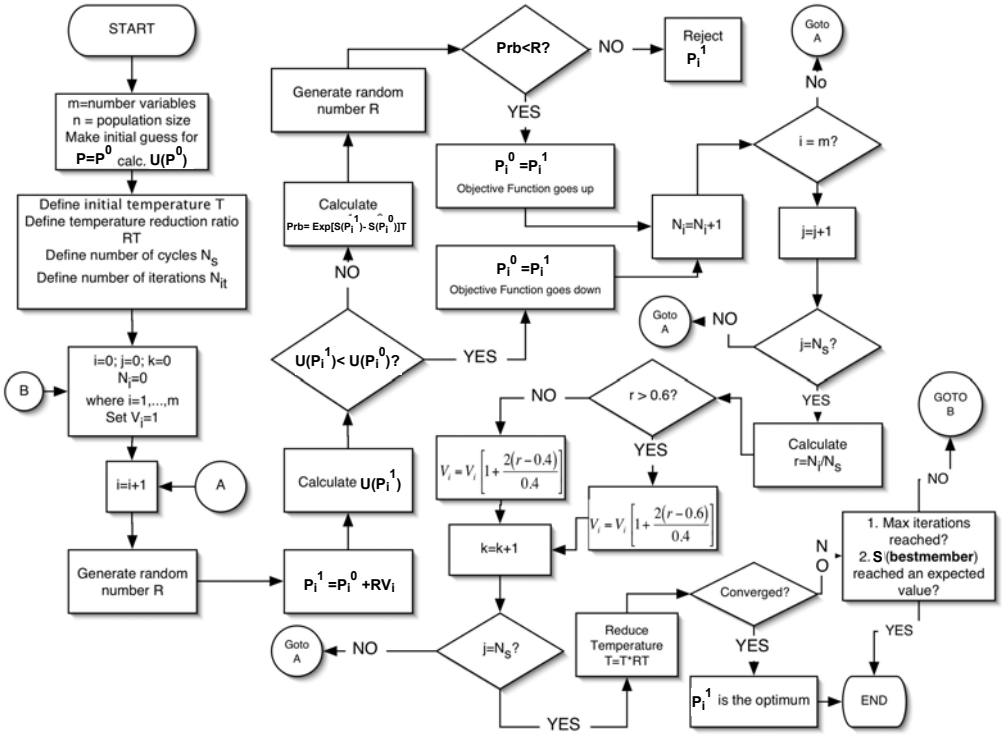


Figure 18 – Iterative procedure for the Simulated Annealing method.

Initially, it randomly chooses a trial point within the step length V (a vector of length n) of the user selected starting point. The function is evaluated at this trial point (P_i^1) and its value is compared to its value at the initial point (P_i^0). In a minimization problem, all downhill moves are accepted and the algorithm continues from that trial point. Uphill moves may also be accepted; the decision is made by the Metropolis [35] criteria. It uses T (temperature) and the size of the downhill move in a probabilistic manner

$$P = e^{\left[S\left(P_i^1\right) - S\left(P_i^0\right)\right]/T} \tag{38}$$

The smaller T and the size of the uphill move are, the more likely that move will be accepted. If the trial is accepted, the algorithm moves on from that point. If it is rejected, another point is chosen for a trial evaluation.

Each element of V is periodically adjusted, so that half of all function evaluations in that direction are accepted. The number of accepted function evaluations is represented by the variable N^i. Thus, the variable r represents the ratio of accepted over total function evaluations for an entire cycle N_s and it is used to adjust the step length V.

A decrease in T is imposed upon the system with the RT variable by using

$$T(i+1) = RT * T(i) \tag{39}$$

where i is the i-th iteration. Thus, as T declines, uphill moves are less likely to be accepted and the percentage of rejections rises. Given the scheme for the selection for V, V falls. Thus, as T declines, V falls and Simulated Annealing focuses upon the most promising area for optimization.

The parameter T is crucial in using Simulated Annealing successfully. It influences V, the step length over which the algorithm searches for optima. For a small initial T, the step length may be too small; thus not enough function evaluations will be performed to find the global optima. To determine the starting temperature that is consistent with optimizing a function, it is worthwhile to run a trial run first. The user should set RT = 1.5 and T = 1.0. With RT > 1.0, the temperature increases and V rises as well. Then, the value of T must be selected that produces a large enough V.

5. Hybrid Optimization Methods [44-59]

The Hybrid Optimization methods are not more than a combination of the deterministic and the evolutionary/stochastic methods, in the sense that they try to use the advantages of each of these methods. The Hybrid Optimization method usually employs an evolutionary/stochastic method to locate a region where the global extreme point is located and then automatically switches to a deterministic method to get to the exact point faster [44].

One of the possible Hybrid Optimization methods encountered in the literature [44-59], called in this paper **H1**, is illustrated in figure 19 [45]. The driven module is very often the Particle Swarm method, which often performs most of the optimization task. When certain percent of the particles find a minima (let us say, some birds already found their best nesting place), the algorithm switches automatically to the Differential Evolution method and the particles (birds) 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 prone to having a global minimum. If there is no improvement on the objective function, this can indicate that this region already contains the global value expected and the algorithm automatically switches to the BFGS method in order to find its location more precisely. In figure 16, the algorithm returns to the Particle Swarm method in order to check if there are no changes in this location and the entire procedure repeats itself. After some maximum number of iterations is performed (e.g., five) the process stops.

In the Particle Swarm method, the probability test of the Simulated Annealing is performed in order to allow the particles (birds) to escape from local minima, although this procedure most often does not make any noticeable improvement in the method.

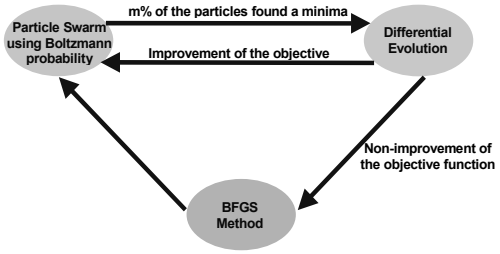


Figure 19 – Global procedure for the Hybrid Optimization method H1.

6. Response Surfaces

From the viewpoint of kernel interpolation/approximation techniques, many response surface methods are based on linear and non-linear regression and other variants of the least square technique. This group of mesh-free methods has been successfully applied to many practical, but difficult problems in engineering that are to be solved by the traditional mesh-based methods.

One of the most popular mesh-free kernel approximation techniques is the one that uses radial basis functions (RBFs). Initially, RBFs were developed for multivariate data and function interpolation. It was found that RBFs were able to construct an interpolation scheme with favorable properties such as high efficiency, good quality and capability of dealing with scattered data, especially for higher dimension problems. A convincing comparison [60] of a RBF based response surface method and a wavelet based artificial neural network method [89] demonstrated superiority of RBF based methods especially for high dimensionality response surfaces.

The use of RBFs followed by collocation, a technique first proposed by Kansa [90], after the work of Hardy [91] on multivariate approximation, is now becoming an established approach. Various applications to problems in mechanics have been made in recent years – see, for example Leitão [92,93].

Kansa's method (or asymmetric collocation) starts by building an approximation to the field of interest (normally displacement components) from the superposition of RBFs (globally or compactly supported) conveniently placed at points in the domain and/or at the boundary.

The unknowns (which are the coefficients of each RBF) are obtained from the approximate enforcement of the boundary conditions as well as the governing equations by means of collocation. Usually, this approximation only considers regular RBFs, such as the globally supported multiquadrics or the compactly supported Wendland functions [94].

There are several other methods for automatically constructing multi-dimensional response surfaces. Notably, a classical book by Lancaster and Salkauskas [95] offer a variety of methods for fitting hypersurfaces of a relatively small dimensionality. Kauffman *et al*. [96] obtained reasonably accurate fits of data by using second order polynomials. Ivakhnenko and his team in Ukraine [97] have published an exceptionally robust methods for fitting non-smooth data points in multi-dimensional spaces. Their method is based on a self-assembly approach where the analytical description of a hypersurface is a multi-level graph of the type “polynomial-of-a-polynomial-of-a-polynomial-of-a-…” and the

basis functions are very simple polynomials [98]. This approach has been used in Indirect Optimization Based Upon Self-Organization (IOSO) [99] commercial optimization software that has been known for its extraordinary speed and robustness.

6.1. The RBF Model Used in this Paper

Let us suppose that we have a function of L variables P_i, $i = 1,\ldots, L$. The RBF model used in this work has the following form

$$S(\mathbf{P}) \cong \xi(\mathbf{P}) = \sum_{j=1}^{N} \alpha_j \phi\left(\left|\mathbf{P} - \mathbf{P}_j\right|\right) + \sum_{k=1}^{M}\sum_{i=1}^{L} \beta_{i,k} q_k(P_i) + \beta_0 \tag{40}$$

where $\mathbf{P}=\{P_1,\ldots,P_i,\ldots,P_L\}$ and $S(\mathbf{P})$ is known for a series of points $\mathbf{P}$. Here, $q_k(P_i)$ is one of the M terms of a given basis of polynomials [100]. This approximation $\xi(\mathbf{P})$ is solved for the α_j and $\beta_{i,k}$ unknowns from the system of N linear equations, subject

$$\begin{aligned} &\sum_{j=1}^{N} \alpha_j q_k(P_1) = 0 \\ &\qquad \vdots \\ &\sum_{j=1}^{N} \alpha_j q_k(P_L) = 0 \end{aligned} \tag{41}$$

$$\sum_{j=1}^{N} \alpha_j = 0 \tag{42}$$

In this work, the polynomial part of equation (40) was taken as

$$q_k(P_i) = P_i^k \tag{43}$$

and the radial basis functions are selected among the following

$$\text{Multiquadrics}: \quad \phi\left(\left|P_i - P_j\right|\right) = \sqrt{\left(P_i - P_j\right)^2 + c_j^2} \tag{44.a}$$

$$\text{Gaussian}: \quad \phi\left(\left|P_i - P_j\right|\right) = \exp\left[-c_j^2\left(P_i - P_j\right)^2\right] \tag{44.b}$$

$$\text{Squared multiquadrics}: \quad \phi\left(\left|P_i - P_j\right|\right) = \left(P_i - P_j\right)^2 + c_j^2 \tag{44.c}$$

$$\text{Cubical Multiquadrics}: \quad \phi\left(\left|P_i - P_j\right|\right) = \left[\sqrt{\left(P_i - P_j\right)^2 + c_j^2}\right]^3 \tag{44.d}$$

with the shape parameter c_j kept constant as $1/N$. The shape parameter is used to control the smoothness of the RBF. Figure 20 shows the influence on its choice for the multiquadrics RBF. From equation (40) one can notice that a polynomial of order M is added to the radial basis function. M was limited to an upper value of 6. After inspecting equations (40)-(43), one can easily check that the final linear system has $[(N+M*L)+1]$ equations. Some tests were made using the cross-product polynomials ($P_i P_j P_k \ldots$), but the improvements of the results were irrelevant. Also, other types of RBFs were used, but no improvement of the interpolation was observed.

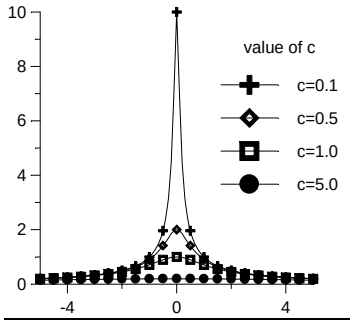


Figure 20 – Influence of the shape parameter

The choice of which polynomial order and which RBF are the best to a specific function, was made based on a cross-validation procedure. Let us supose that we have N_{TR} training points, which are the locations on 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 N_{TR1} and N_{TR2}. The equations (40)-(42) are solved for a polynomial of order zero and for the RBF expression given by equations (44) using the subset N_{TR1}. Then, the value of the interpolated function is checked against the known value of the function for the subset N_{TR2} and the error is recorded as

$$RMS_{N_{TR1},M=0,RBF_1} = \sum_{i=1}^{N_{TR2}} \left[S(P_i) - \xi(P_i)\right]^2 \tag{45}$$

Then, the same procedure is made, using the subset N_{TR2} to solve equations (40)-(42) and the subset N_{TR1} to calculate the error as

$$RMS_{N_{TR2},M=0,RBF_1} = \sum_{i=1}^{N_{TR1}} \left[S(P_i) - \xi(P_i)\right]^2 \tag{46}$$

Finally, the total error for the polynomial of order zero and the RBF expression given by equations (44) is obtained as

$$RMS_{M=0,RBF_1} = \sqrt{RMS_{N_{TR1},M=0,RBF_1} + RMS_{N_{TR2},M=0,RBF_1}} \tag{47}$$

This procedure is repeated for all polynomial orders, up to $M = 6$ and for each one of the RBF expressions given by equations (44). 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 all test cases analysed in this paper.

6.2. Performance Measurements

In accordance with having multiple metamodeling criteria, the performance of each metamodeling technique is measured from the following aspects [101].

- Accuracy – the capability of predicting the system response over the design space of interest.
- Robustness – the capability of achieving good accuracy for different problem types and sample sizes.
- Efficiency – the computational effort required for constructing the metamodel and for predicting the response for a set of new points by metamodels.
- Transparency – the capability of illustrating explicit relationships between input variables and responses.
- Conceptual Simplicity – ease of implementation. Simple methods should require minimum user input and be easily adapted to each problem.

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, three different metrics are used: R Square, Relative Average Absolute Error (RAAE), and Relative Maximum Absolute Error (RMAE) [101].

a) R Square (R2)

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{MSE}{\text{variance}} \tag{48}$$

where $\hat{y}_i$ is the corresponding predicted value for the observed value y_1; $\bar{y}$ 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)

$$RAAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n * STD} \tag{49}$$

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

c) Relative Maximum Absolute Error (RMAE)

$$RMAE = \frac{\max\left(|y_1 - \hat{y}_1|, |y_2 - \hat{y}_2|, ..., |y_n - \hat{y}_n|\right)}{STD} \tag{50}$$

Large RMAE indicates large error in one region of the design space even though the overall accuracy indicated by R2 and RAAE can be very good. Therefore, a small RMAE is preferred. However, since this metric cannot show the overall performance in the design space, it is not as important as R2 and RAAE.

Although the R2, RAAE and RMAE are useful to ascertain the accuracy of the interpolation, they can fail in some cases. For the R2 metric, for example, if one of the testing points has a huge deviation of the exact value, such discrepancy might affect the entire sum appearing on equation (48) and, even if all the other testing points are accurately interpolated. Similary, the R2 result can be very bad. For this reason, we also calculate the percentage deviation of the exact value of each testing point. Such deviations are collected according to 6 ranges of errors: 0-10%; 10-20%; 20-50%; 50-100%; 100-200%; >200%. Thus, a interpolation that has all testing points within the interval of 0 to 10% of relative error might be considered good in comparison to another one where the points are all spread along the intervals from 10% to 200%.

6.3. Response Surface Test Cases

In order to show the accuracy of the RBF model presented, 296 test cases were used, representing linear and non-linear problems with up to 100 variables. Such problems were selected from a collection of 395 problems (actually 296 test cases), proposed by Hock and Schittkowski [101] and Schittkowski [102]. Figure 21 shows the number of variables of each one of the problems. Note that there are 395 problems, but some of them were not used.

Three methodologies were used to solve the linear algebraic system resulting from equations (40)-(42): LU decomposition, SVD and the Generalized Minimum Residual (GMRES) iterative solver. When the number of equations was small (less than 40), the LU solver was used. However, when the number of variables increased over 40, the resulting matrix becomes too ill-conditioned and the SVD solver had to be used. For more than 80 variables, the SVD solver became too slow. Thus, the GMRES iterative method with the Jacobi preconditioner was used for all test cases.

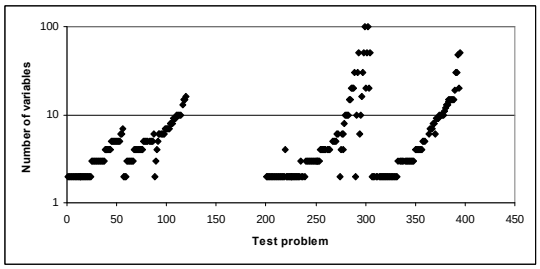


Figure 21 – Number of variables for each problem considered

In order to verify the accuracy of the interpolation over different number of training points, three sets were defined. Also, the number of testing points varied, according to the number of training points. Table 1 presents these three sets, based on the number of dimensions (variables) *L* of the problem

Table 1 – Number of training and testing points

	Number of training points	Number of testing points
Scarce set	3 *L*	300 *L*
Small set	10 *L*	1000 *L*
Medium set	50 *L*	5000 *L*

Figure 22 shows the R2 metric for all test cases, using the scarce set of training points. It can be noticed that the results are all spread from 0 (completely inadequate interpolation) to 1 (very accurate interpolation). However, even for this very small number of training points, most cases have an excelent interpolation, with R2 = 1.

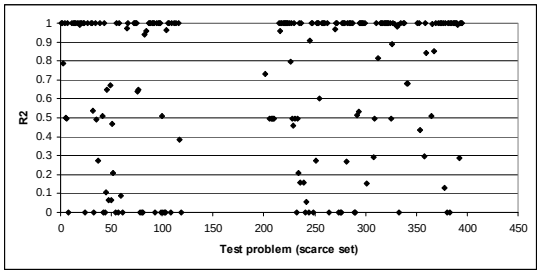


Figure 22 – R2 metric for the scarse set of training points

Figure 23 shows the CPU time required to interpolate each test function, using the scarce set of training points. For most of the cases, the CPU time was lesser than one second, using an AMD Opteron 1.6 GHz processor and 1GB Registered ECC DDR PC-2700 RAM. In fact, the highest dimensional test cases, which had 100 variables, required only 100 seconds to be interpolated.

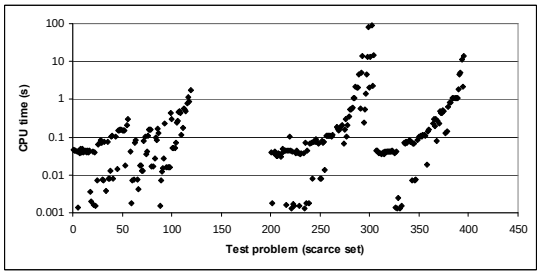


Figure 23 – CPU time for the scarse set of training points

Although the R2 might indicate some performance behaviour of the interpolation function, we decided to use a different measure of the accuracy. Figure 24 shows the percentage of testing points having errors less than 10%, against the percentage of all 296 test cases, for the scarce set of testing points. Thus, from this figure, it can be noticed that for more than 40% of all test functions, the relative errors were less than 10%. This is a very good result, considering the extremely small number of training points used in the scarce set.

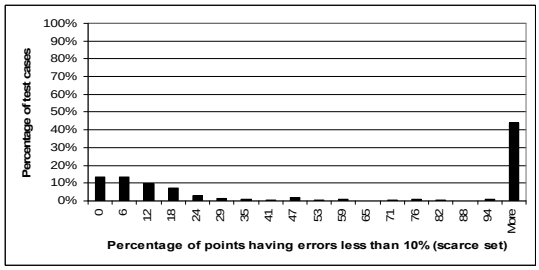


Figure 24 – Testing points with less than 10% error, for the scarce set of training points

Figure 25 shows the R2 metric for the small set of training points. Comparing with figure 22, it can be seen that the points move toward the value of R2 = 1.0, showing that the accuracy of the interpolation gets better when the number of training points increase.

Figure 26 shows the CPU time required for all test cases, when the small number of training points is used. Although the test case with 100 variables requires almost 1000 seconds, in almost all test cases the CPU time is low.

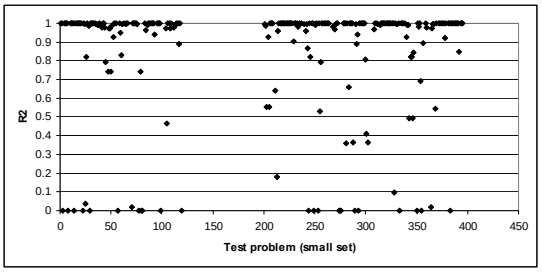


Figure 25 – R2 metric for the small set of training points

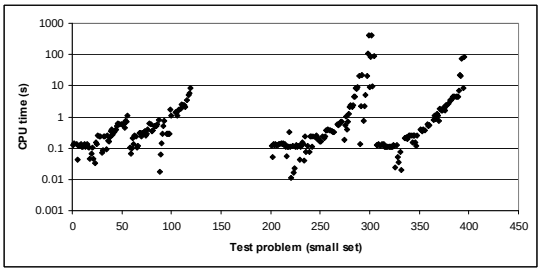


Figure 26 – CPU time for the small set of training points

Figure 27 shows the percentage of points having errors lower than 10%. Comparing with figure 24, one can see that increasing the number of training points from 3*L* (scarce set) to 10*L* (small set), the number of testing points having less than 10% of relative error for all 296 test cases increase from approximately 45% to approximately 55%, showing a very good interpolation, even for a not so large number of training points.

Finally, figures 28-30 show the results when a medium set of training points are used.

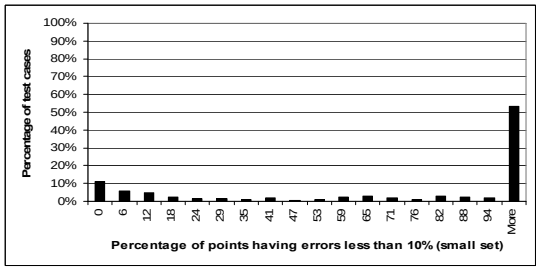


Figure 27 – Testing points with less than 10% error, for the small set of training points

From figure 28, one can notice that the majority of the test cases have the R2 metric close to 1.0, indicating a very good interpolation, for a not so large CPU time, as it can be verified at figure 29. From figure 30, the number of testing points having errors less than 10% for all 296 test cases increases to approximately 75% when a medium (50 *L*) number of training points is used. This indicates that such interpolation can be used as a meta model in a optimization task, where the objective function takes too long to be calculated. Thus, instead of optimizing the original function, a interpolation can be used, reducing significativally the computational time.

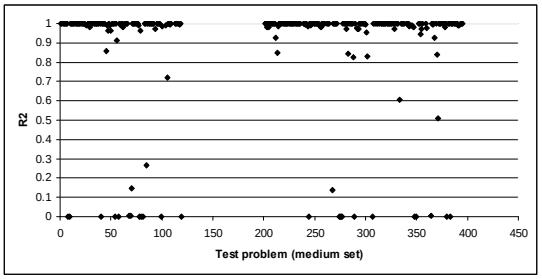


Figure 28 – R2 metric for the medium set of training points

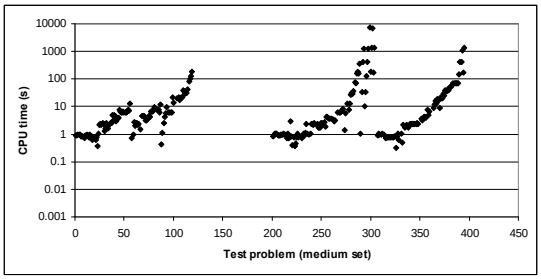


Figure 29 – CPU time for the medium set of training points

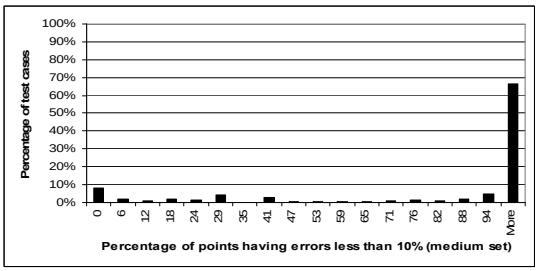


Figure 30 – Testing points with less than 10% error, for the medium set of training points

7. Hybrid Methods with Response Surfaces and Examples

Once the response surface methodology and the hybrid optimizer idea were presented, we will combine , in this section, both of them in a more powerful method. Such method, called hybrid optimizer ***H2*** [48], is quite similar to the ***H1*** presented in section 5, except by the fact that is uses a response surface method at some point of the optimization task. The global procedure is illustrated in figure 31. It can be seen from this figure that after a certain number of objective functions were calculated, all this information was used to obtain a response surface. Such a response surface is then optimized using the same proposed hybrid code defined in the ***H1*** optimizer so that it fits the calculated values of the objective function as closely as possible. New values of the objective function are then obtained very cheaply by interpolating their values from the response surface

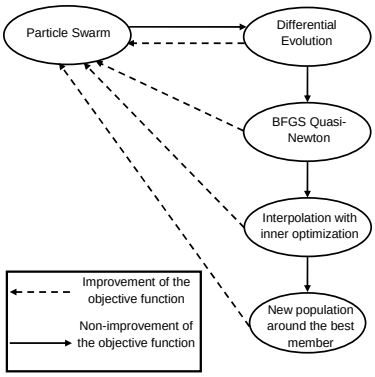


Figure 31 – Global procedure for the hybrid optimization method ***H2***

In figure 31, if the BFGS cannot find any better solution, the algorithm uses a radial basis function interpolation scheme to obtain a response surface and then optimizes such response surface using the same hybrid algorithm proposed. When the minimum value of this response surface is found, the algorithm checks to see if it is also a solution of the original problem. Then, if there is no improvement of the objective function, the entire population is eliminated and a new population is generated around the best value obtained so far. The algorithm returns to the particle swarm method in order to check if there are no changes in this location and the entire procedure repeats itself. After a specified maximum number of iterations is performed (*e.g.*, five) the process stops.

An even more efficient algorithm, which will be called ***H3*** is an extension of the previous ones. The global procedure is enumerated below:

1. Generate an initial population, using the real function (not the interpolated one) $f(\mathbf{P})$. Call this population Π_{real}.

2. Determine the individidual that has the minimum value of the objective function, over the entire population $\mathbf{\Pi}_{real}$ and call this individual $\mathbf{P}_{best}$.
3. Determine the individual that is more distante from the $\mathbf{P}_{best}$, over the entire population $\mathbf{\Pi}_{real}$. Call this individual $\mathbf{P}_{far}$.
4. Generate a response surface, with the methodology at section 6, using the entire population $\mathbf{\Pi}_{real}$ as training points. Call this function $g(\mathbf{P})$.
5. Optimize the interpolated function $g(\mathbf{P})$ using the hybrid optimizer ***H1***, defined in section 5, and call the optimum variable of the interpolated function as $\mathbf{P}_{int}$. During the generation of the internal population to be used in the ***H1*** optimizer, consider the upper and lower bounds limits as the minimum and maximum values of the population $\mathbf{\Pi}_{real}$ in order to not extrapolate the response surface.
6. If the real objective function $f(\mathbf{P}_{int})$ is better than all objective function of the population $\mathbf{\Pi}_{real}$, replace $\mathbf{P}_{far}$ by $\mathbf{P}_{int}$. Else, generate a new individual, using the sobol pseudo-random generator within the upper and lower bounds of the variables, and replace $\mathbf{P}_{far}$ by this new individual.
7. If the optimimum is achieved, stop the procedure. Else, return to step 2.

From the sequence above, one can notice that the number of times that the real objective function $f(\mathbf{P})$ is called is very small. Also, from step 6, one can see that the space of search is reduced at each iteration. When the response surface $g(\mathbf{P})$ is no longer capable to found a minimum, a new call to the real function $f(\mathbf{P})$ is made to generate a new point to be included in the interpolation. Since the CPU time to calculate the interpolated function is very small, the maximum number of iterations of the ***H1*** optimizer can be very large (e.g., 1000 iterations).

The hybrid optimizer ***H3*** was compared against the optimizer ***H1***, ***H2*** and the commercial code IOSO 2.0 for some standard test functions. The first test function was the Levy #9 function [103], which has 625 local minima and 4 variables. Such function is defined as

$$S(\mathbf{P}) = \sin^2(\pi - z_1) + \sum_{i=1}^{n-1}(z_i - 1)^2\left[1 = 10\sin^2(\pi\, z_{i+1})\right] + (z_4 - 1)^2 \tag{51}$$

where

$$z_i = 1 + \frac{P_i - 1}{4}, (i = 1,4) \tag{52}$$

The function is defined within the interval $-10 \le \mathbf{P} \le 10$ and its minimum is $S(\mathbf{P}) = 0$ for $\mathbf{P} = 1$. Figure 32 shows the optimization history of the IOSO, ***H1***, ***H2*** and ***H3*** optimizers. Since the ***H1***, ***H2*** and ***H3*** optimizers are based on random number generators (because the Particle Swarm module), we present the best and worst estimatives for these three optimizers.

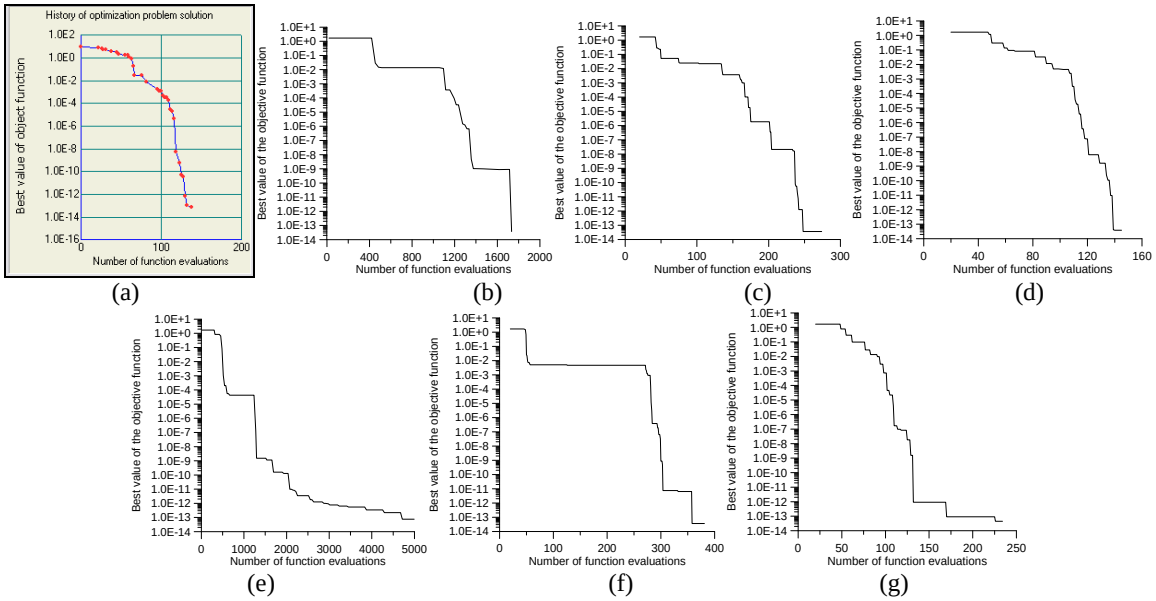


Figure 32 – Optimization history of the Levy #9 function for the (a)IOSO, (b)***H1-best***, (c)***H2-best***, (d)***H3-best***, (e)***H1-worst***, (f)***H2-worst*** and (g)***H3-worst*** optimizers

From figure 32, it can be seen that the performance of the ***H3*** optimizer is very close to the IOSO commercial code. The ***H1*** code is the worst and the ***H2*** optimizer also has a reasonable good performance. It is interesting to note that the ***H1*** code is the only one that doesn't have a response surface model implemented.

The second function tested was the Griewank function [59], which is defined as

$$S(\mathbf{P}) = \sum_{i=1}^{n} \frac{P_i^2}{4000} - \prod_{i-1}^{n} \cos\left(\frac{P_i}{\sqrt{i}}\right) + 1 \quad ; \qquad P_i \in]-600,600[\quad ,(i = 1,2) \tag{53}$$

The global minima for this function is located at $\mathbf{P} = 0$ and is $S(\mathbf{P}) = 0$. The function has an incredibly number of local minima, making the optimization task quite difficult.

Figure 33 shows the optimization history of the IOSO, ***H1***, ***H2*** and ***H3*** optimizers. Again, the best and worst results for ***H1***, ***H2*** and ***H3*** are presented.

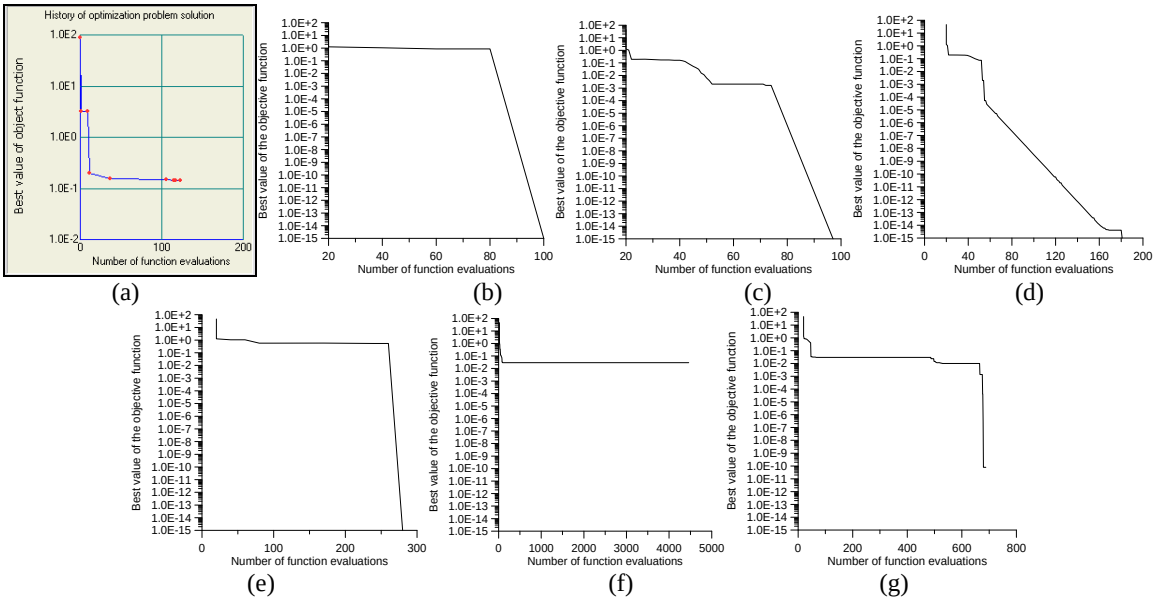


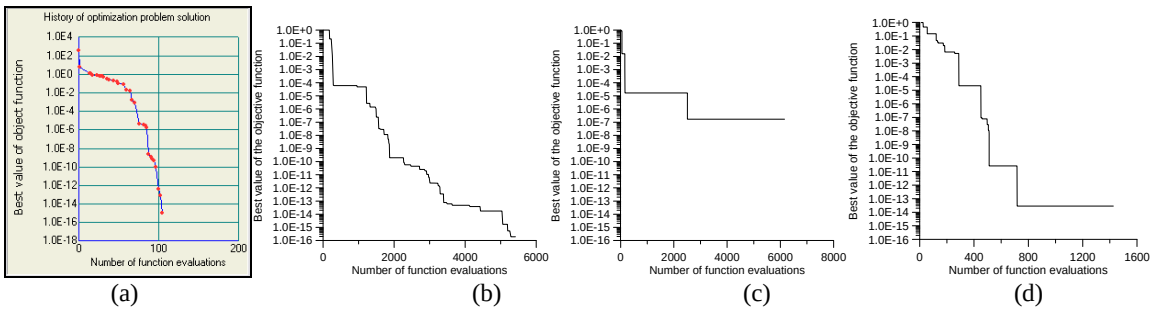
Figure 33 – Optimization history of the Griewank function for the (a)IOSO, (b)***H1-best***, (c)***H2-best***, (d)***H3-best***, (e)***H1-worst***, (f)***H2-worst*** and (g)***H3-worst*** optimizers

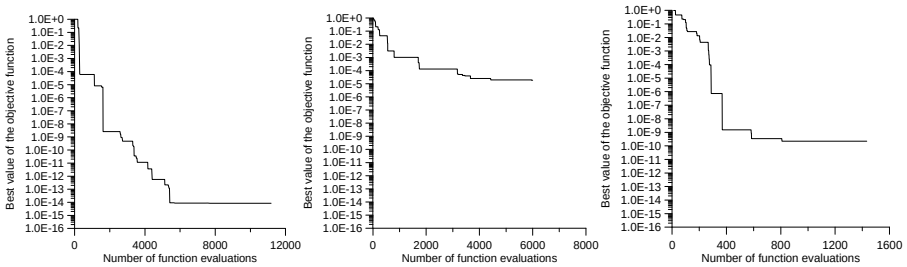
From this figure, it is clear that the ***H1***, ***H2*** and ***H3*** optimizers are much better than the IOSO commercial code. The ***H1*** code was the best, while the ***H2*** sometimes stopped at some local minima. The worst result of the ***H3*** optimizer was, however, better than the result obtained by IOSO. It is worth to notice that the, with more iterations, the ***H3*** code could reach the minimum of the objective function, even for the worst result.

The next test function implemented was the Rosenbrook function [104], which is defined as

$$S(P_1, P_2) = 100(P_2 - P_1^2)^2 + (1 - P_1)^2 \tag{54}$$

The function is defined within the interval $-10 \leq \mathbf{P} \leq 10$ and its minimum is $S(\mathbf{P}) = 0$ for $\mathbf{P} = 1$. Figure 34 shows the optimization history of the IOSO, ***H1***, ***H2*** and ***H3*** optimizers.





(e) (f) (g)

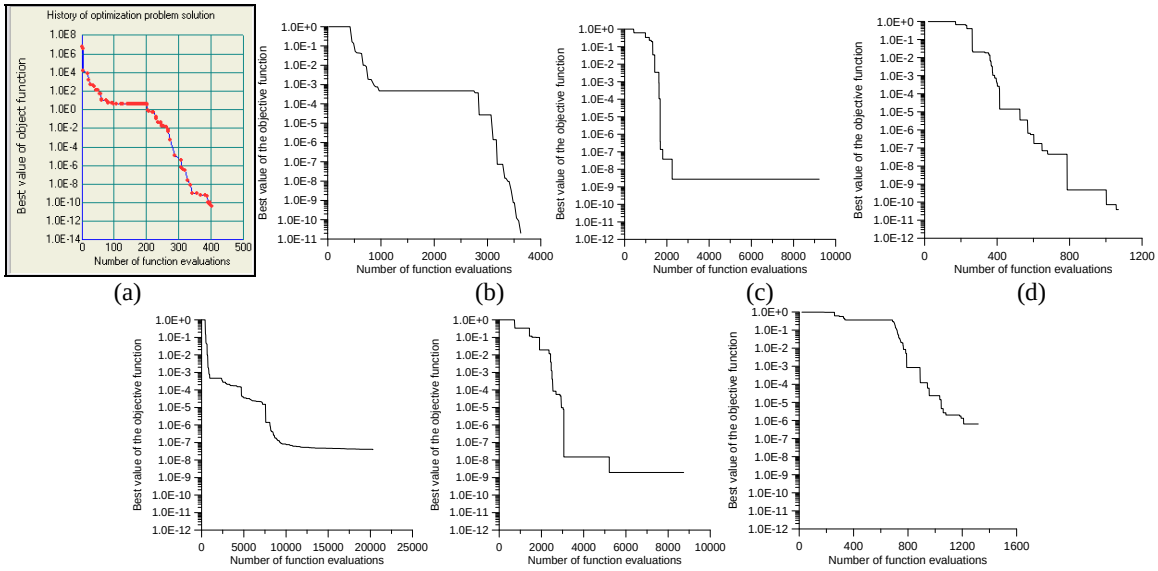
Figure 34 – Optimization history of the Rosenbrook function for the (a)IOSO, (b)***H1-best***, (c)***H2-best***, (d)***H3-best***, (e)***H1-worst***, (f)***H2-worst*** and (g)***H3-worst*** optimizers

For this function, which is almost flat close to the global minima, the IOSO code was the one with the best performance, followed by the ***H3*** optimizer. The ***H2*** performed very bad and the ***H1*** was able to get close to the minimum, but with a huge number of objective function calculations. When looking at the ***H3*** results, the final value of the objective function differ by some orders of magnitude. However, the optimum solution obtained with this new optimizer was $P_1 = 0.9996$ and $P_2 = 0.9992$, while the IOSO obtained $P_1 = 1.0000$ and $P_2 = 1.0000$. Thus the relative error among the variables was less than 0.01%, indicating that despite of the discrepancy among the final value of the objective function, the ***H3*** code was able to recover the value of the optimum variables with a neglectable relative error.

The last test function analyzed was the Mielle-Cantrel function [105], which is defined as

$$S(\mathbf{P}) = \left[\exp^{(P_1 - P_2)}\right]^4 + 100(P_2 - P_3)^6 + \arctan^4(P_3 - P_4) + P_1^2 \tag{55}$$

The function is defined within the interval $-10 \leq \mathbf{P} \leq 10$ and its minimum is $S(\mathbf{P}) = 0$ for $P_1 = 0$ and $P_2 = P_3 = P_4 = 1$. Figure 35 shows the optimization history of the IOSO, ***H1***, ***H2*** and ***H3*** optimizers. Again, the best and worst results for ***H1***, ***H2*** and ***H3*** are presented.



(a) (b) (c) (d)

(e) (f) (g)

Figure 35 – Optimization history of the Griewank function for the (a)IOSO, (b)***H1-best***, (c)***H2-best***, (d)***H3-best***, (e)***H1-worst***, (f)***H2-worst*** and (g)***H3-worst*** optimizers

For this function, the IOSO code was the best, followed by the ***H3***. The ***H2*** code performed very bad again the the ***H1*** was able to get to the global mininum after a huge numbe of objective function calculations. As ocorred with the Rosenbrook function, in spite of the ***H3*** result for the objective function differ from the IOSO code, the final values of the variables were P_1=4.0981x10^{-8}, P_2=0.9864, P_3=0.9688 and P_4=0.9626 for the ***H3*** optimizer and P_1=-0.1216x10^{-5}, P_2=1.002, P_3=0.9957 and P_4=0.9962 for the IOSO code.

8. Conclusion

In this paper we presented some basic concepts related to deterministic and heuristic methods, applied to single objective optimization. Three different hybrid methods were also presented, as well as a powerfull response surface methodology. The combination of the techniques presented here can be used in very complex engineering problems, which demand thousands of objective functions calculations.

Aknowledgements

This work was partially funded by CNPq, CAPES (agencies for the fostering of science from the Brazilian Ministry of Science and Education) and FAPERJ (agency for the fostering of science from the Rio de Janeiro State). The authors are also very thankful to Prof. Alain J. Kassab from University of Central Florida for his suggestions (during IPDO-2007 in Miami) on how to choose the best shape parameter for RBF approximations. The first author is very gratefull for the hospitality of George and Ellen Dulikravich during his staying in Miami in several periods of the years 2006, 2007, 2008 and 2009.

9. References

[1] A. N. Tikhonov and V. Y. Arsenin, Solution of Ill-Posed Problems, Winston & Sons, Washington, DC (1977).
[2] J. V. Beck and K. J. Arnold, Parameter Estimation in Engineering and Science, Wiley Interscience, New York (1977).
[3] O. M. Alifanov, Inverse Heat Transfer Problems, Springer-Verlag, New York (1994).
[4] J. V. Beck, B. Blackwell and C. R. St. Clair, Inverse Heat Conduction: Ill-Posed Problems, Wiley Interscience, New York (1985).
[5] O. M. Alifanov, E. Artyukhin and A. Rumyantsev, Extreme Methods for Solving Ill-Posed Problems with Applications to Inverse Heat Transfer Problems, Begell House, New York (1995).
[6] G. S. Dulikravich and T. J. Martin, Inverse Shape and Boundary Condition Problems and Optimization in Heat Conduction, Chapter 10 in Advances in Numerical Heat Transfer, 1, 381-426, W. J. Minkowycz and E. M. Sparrow (eds.), Taylor and Francis (1996).
[7] M. N. Ozisik and H. R. B. Orlande, Inverse Heat Transfer: Fundamentals and Applications, Taylor and Francis, New York (2000).
[8] K. Kurpisz and A. J. Nowak, Inverse Thermal Problems, WIT Press, Southampton, UK (1995).
[9] K. Woodbury, Inverse Engineering Handbook, CRC Press, Boca Raton, FL (2002).
[10] D. A. Murio, The Mollification Method and the Numerical Solution of Ill-Posed Problems, Wiley Interscience, New York (1993).
[11] D. M. Trujillo and H. R. Busby, Practical Inverse Analysis in Engineering, CRC Press, Boca Raton, FL (1997).
[12] J. V. Beck, Sequential Methods in Parameter Estimation, 3rd International Conference on Inverse Problems in Engineering, Tutorial Session, Port Ludlow, WA (1999).
[13] M. J. Colaço and H. R. B. Orlande, Inverse Forced Convection Problem of Simultaneous Estimation of Two Boundary Heat Fluxes in Irregularly Shaped Channels, Numerical Heat Transfer, Part A, 39, 737-760 (2001).
[14] M. J. Colaço and H. R. B. Orlande, Inverse Natural Convection Problem of Simultaneous Estimation of Two Boundary Heat Fluxes in Irregular Cavities, International J. of Heat and Mass Transfer, 47, 1201–1215 (2004).
[15] M. J. Colaço and H. R. B. Orlande, A Comparison of Different Versions of the Conjugate Gradient Method of Function Estimation. Numerical Heat Transfer. Part A, Applications, 36 (2), 229-249 (1999).
[16] J. W. Daniel, The Approximate Minimization of Functionals, Prentice-Hall, Englewood Cliffs (1971).
[17] Y. Jaluria, Design and Optimization of Thermal Systems, McGraw Hill (1998).
[18] W. F. Stoecker, Design of Thermal Systems, McGraw Hill (1989).
[19] A. D. Belegundu and T. R. Chandrupatla, Optimization Concepts and Applications in Engineering, Prentice Hall (1999).
[20] R. Fletcher, Practical Methods of Optimization, John Wiley & Sons (2000).
[21] M. J. D. Powell, Restart Procedures for the Conjugate Gradient Method, Mathematical Programming, 12, 241-254 (1977).
[22] R. Fletcher and C. M. Reeves, Function Minimization by Conjugate Gradients, Comput. Journal, 7, 149-154 (1964).
[23] M. R. Hestenes and E. Stiefel, Method of Conjugate Gradients for Solving Linear Systems, J. Res. Nat. Bur. Standards Sect. B, 49, 409-436 (1952).
[24] E. Polak, Computational Methods in Optimization, Academic Press, New York (1971).
[25] E. M. L. Beale, A Derivation of Conjugate Gradients, Numerical Methods for Nonlinear Optimization, F. A. Lootsma (editor), 39-43 (1972).
[26] W. C. Davidon, Variable Metric Method for Minimization, Argonne Natl. Lab., ANL-5990 (1959).
[27] R. Fletcher and M. J. D. Powell, A Rapidly Convergent Descent Method for Minimization, Computer J., 6, 163-168 (1963).

[28] C. G. Broyden, A Class of Methods for Solving Nonlinear Simultaneous Equations, Math. Comp., 19, 577-593 (1965).
[29] C. G. Broyden, Quasi-Newton Methods and Their Applications to Function Minimization, Math. Comp., 21, 368-380 (1967).
[30] K. Levenberg, A Method for the Solution of Certain Non-linear Problems in Least Squares, Quart. Appl. Math., 2, 164-168 (1944).
[31] D. W. Marquardt, An Algorithm for Least Squares Estimation of Nonlinear Parameters, J. Soc. Ind. Appl. Math, 11, 431-441 (1963).
[32] Y. B. Bard, Nonlinear Parameter Estimation, Acad. Press, New York (1974).
[33] J. Dennis and R. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice Hall (1983).
[34] J. J. Moré, The Levenberg-Marquardt Algorithm: Implementation and Theory, in: Numerical Analysis, Lecture Notes in Mathematics, Vol. 630, G. A. Watson (ed.), Springer-Verlag, Berlin, 105-116 (1977).
[35] A. Corana, M. Marchesi, C. Martini and S. Ridella, Minimizing Multimodal Functions of Continuous Variables with the 'Simulated Annealing Algorithm', ACM Transactions on Mathematical Software, 13, 262-280 (1987).
[36] W. L. Goffe, G. D. Ferrier and J. Rogers, Global Optimization of Statistical Functions with Simulated Annealing, Journal of Econometrics, 60, 65-99 (1994).
[37] R. Storn and K. V. Price, Minimizing the Real Function of the ICEC'96 Contest by Differential Evolution, IEEE Conf. on Evolutionary Computation, 842-844 (1996).
[38] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Pub Co (1989).
[39] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons (2002).
[40] J. Kennedy and R. C. Eberhart, Particle Swarm Optimization, Proceedings of the 1995 IEEE International Conference on Neural Networks, 4, 1942-1948 (1995).
[41] J. Kennedy, Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance, Proceedings of the 1999 Congress of Evolutionary Computation, IEEE Press, 3, 1931-1938 (1999).
[42] S. Naka, T. G. Yura and T. Fukuyama, Practical Distribution State Estimation Using Hybrid Particle Swarm Optimization, Proceedings IEEE Power Engineering Society, Winter Meeting, Columbus, Ohio, January 28-February 1st (2001).
[43] R. Eberhart, Y. Shi and J. Kennedy, Swarm Intelligence, Morgan Kaufmann (2001).
[44] G. S. Dulikravich, T. J. Martin, B. H. Dennis and N. F. Foster, Multidisciplinary Hybrid Constrained GA Optimization, Invited lecture, Chapter 12 in EUROGEN'99 - Evolutionary Algorithms in Engineering and Computer Science: Recent Advances and Industrial Applications, (eds: K. Miettinen, M. M. Makela, P. Neittaanmaki and J. Periaux), John Wiley & Sons, Jyvaskyla, Finland, May 30-June 3, 1999, pp. 231-260 (1999).
[45] M. J. Colaço, G. S. Dulikravich, H. R. B. Orlande and T. J. Martin, Hybrid Optimization with Automatic Switching Among Optimization Algorithms. In: W. Annicchiarico; J. Periaux; M. Cerrolaza; G. Winter. (Org.). Evolutionary Algorithms and Intelligent Tools in Engineering Optimization. : WIT Press/Computational Mechanics, 92-118 (2005).
[46] M. J. Colaço, G. S. Dulikravich and D. Sahoo, A Response Surface Method Based Hybrid Optimizer. Inverse Problems in Science & Engineering, 16, 717-742 (2008).
[47] G. S. Dulikravich, I. N. Egorov and M. J. Colaço, Optimizing Chemistry of Bulk Metallic Glasses for Improved Thermal Stability. Modelling and Simulation in Materials Science and Engineering, 16, 075010 (2008).
[48] M. J. Colaço and G. S. Dulikravich, Solidifcation of Double-Diffusive Flows Using Thermo-Magneto-Hydrodynamics and Optimization. Materials and Manufacturing Processes, 22, 594-606 (2007).
[49] P. M. P. Silva, H. R. B. Orlande, M. J. Colaço, P. S. Shiakolas and G. S. Dulikravich, Identification and Design of a Source Term in a Two-Region Heat Conduction Problem. Inverse Problems in Science & Engineering, 15, 661-677 (2007).
[50] M. J. Colaço, H. R. B. Orlande and G. S. Dulikravich, Inverse and Optimization Problems in Heat Transfer. Journal of the Brazilian Society of Mechanical Sciences and Engineering, Brasil, 28 (1), 1-24 (2006).
[51] M. J. Colaço and G. S. Dulikravich, A Multilevel Hybrid Optimization of Magnetohydrodynamic Problems in Double-Diffusive Fluid Flow. Journal of Physics and Chemistry of Solids, 67, 1965-1972 (2006).
[52] O. Wellele, H. R. B. Orlande, N. J. Ruberti, M. J. Colaço and A. Delmas, Coupled Conduction-Radiation in Semi-Transparent Materials at High Temperatures. Journal of Physics and Chemistry of Solids, 67, 2230-2240 (2006).
[53] G. S. Dulikravich and M. J. Colaço, Convective Heat Transfer Control Using Magnetic and Electric Fields. Journal of Enhanced Heat Transfer, 13, 2, 139-155 (2006).
[54] M. J. Colaço, G. S. Dulikravich and T. J. Martin, Control of Unsteady Solidification via Optimized Magnetic Fields. Materials and Manufacturing Processes, 20(3), 435-458 (2005).
[55] G. S. Dulikravich, M. J. Colaço, B. H. Dennis, T. J. Marting, I. N. Egorov and S. Lee, Optimization of Intensities, and Orientations of Magnets Controlling Melt Flow During Solidification. Materials and Manufacturing Processes, 19(4), 695-718 (2004).

[56] M. J. Colaço, G. S. Dulikravich and T. J. Martin, Optimization of Wall Electrodes for Electro-Hydrodynamic Control of Natural Convection During Solidification. Materials and Manufacturing Processes, 19 (4), 719-736 (2004).
[57] G. S. Dulikravich, M. J. Colaço, T. J. Martin and S. Lee, An Inverse Method Allowing User-Specified Layout of Magnetized Micro-Fibers in Solidifying Composites. Journal of Composite Materials, UK, 37 (15), 1351-1365 (2003).
[58] M. J. Colaço and H. R. B. Orlande, Inverse Problem of Simultaneous Estimation of Two Boundary Heat Fluxes in Parallel Plate Channels. Journal of the Brazilian Society of Mechanical Engineering, XXIII (2), 201-215 (2001).
[59] R. S. Padilha, H. F. S. Santos, M. J. Colaço and M. E. C. Cruz, Single and Multi-Objective Optimization of a Cogeneration System Using Hybrid Algorithms, Heat Transfer Engineering, 30, 261-271 (2009).
[60] M. J. Colaço, G. S. Dulikravich and D. Sahoo, A Comparison of Two Methods for Fitting High Dimensional Response Surfaces, Proc. of International Symposium on Inverse Problems, Design and Optimization (IPDO-2007), (eds.: Dulikravich, G. S., Orlande, H. R. B., Tanaka, M. and Colaco, M. J.), Miami Beach, Florida, U.S.A., April 16-18 (2007).
[61] P. C. Sabatier (ed.), Applied Inverse Problems, Springer Verlag, Hamburg (1978).
[62] E. Hensel, Inverse Theory and Applications for Engineers, Prentice Hall, New Jersey (1991).
[63] A. M. Denisov, Elements of the Theory of Inverse Problems, VSP, Netherlands (1999).
[64] A. G. Yagola, I. V. Kochikov, G. M. Kuramshina and Y. A. Pentin, Inverse Problems of Vibrational Spectroscopy, VSP, Netherlands (1999).
[65] A. G. Ramm, P. N. Shivakumar and A. V. Strauss (eds.), Operator Theory and Applications, Amer. Math. Soc., Providence (2000).
[66] V. A. Morozov, Methods for Solving Incorrectly Posed Problems, Springer Verlag, New York (1984).
[67] J. P. Zubelli, An Introduction to Inverse Problems: Examples, Methods and Questions, Institute of Pure and Applied Mathematics, Rio de Janeiro, Brazil (1999).
[68] A. Kirsch, An Introduction to the Matheamtical Theory of Inverse Problems, Applied Mathematical Sciences, vol. 120, Springer (1996).
[69] V. Isakov, Inverse Problems for Partial Differential Equations, Applied Mathematical Sciences, 127, Springer (1998).
[70] J. Hadamard, Lectures on Cauchy's Problem in Linear Differential Equations, Yale University Press, New Haven, CT (1923).
[71] J. W. Daniel, The Approximate Minimization of Functionals, Prentice-Hall, Englewood Cliffs (1971).
[72] R. L. Fox, Optimization Methods for Engineering Design, Addison-Wesley Publishing Company (1971).
[73] G. S. Dulikravich and T. J. Martin, Inverse Design of Super-Elliptic Cooling Passages in Coated Turbine Blade Airfoils, AIAA Journal of Thermophysics and Heat Transfer, 8(2), 288-294, April-June (1994).
[74] C. de Boor, A Practical Guide to Splines, Springer-Verlag, New York (1978).
[75] M. A. Oliver and R. Webster, Kriging: A Method of Interpolation for Geographical Information System, Int. J. Geographical Information Systems, 4(3), 313-332 (1990).
[76] O. M. Alifanov, Solution of an Inverse Problem of Heat Conduction by Iteration Methods, Journal of Engineering Physics, 26 (4), 471-475 (1974).
[77] E. A. Artyukhin, Iterative Algorithms for Estimating Temperature-Dependent Thermophysical Characteristics, 1st Int. Conf. on Inverse Problems in Engineering: Theory and Practice, 101-108., N. Zabaras, K. Woodburry and M. Raynaud (editors), Palm Coast, FL, June (1993).
[78] Y. Jarny, M. N. Ozisik and J. P. Bardon, A General Optimization Method Using Adjoint Equation for Solving Multidimensional Inverse Heat Conduction, Int. J. Heat Mass Transfer, 34 (11), 2911-2919 (1991).
[79] B. Truffart, J. Jarny and D. Delaunay, A General Optimization Algorithm to Solve 2-D Boundary Inverse Heat Conduction Problems Using Finite Elements, 1st Int. Conf. on Inverse Problems in Engineering: Theory and Practice, 53-60, N. Zabaras, K. Woodburry and M. Raynaud (editors), Palm Coast, FL, June (1993).
[80] L. B. Dantas and H. R. B. Orlande, A Function Estimation Approach for Determining Temperature-Dependent Thermophysical Properties, Inverse Problems in Engineering, 3, 261-279 (1996).
[81] H. A. Machado and H. R. B. Orlande, Inverse Analysis for Estimating the Timewise and Spacewise Variation of the Wall Heat Flux in a Parallel Plate Channel, International Journal for Numerical Methods for Heat & Fluid Flow, 7(7), 696-710 (1997).
[82] H. R. B. Orlande, M. J. Colaço and A. A. Malta, Estimation of the Heat Transfer Coefficient in the Spray Cooling of Continuously Cast Slabs, National Heat Transfer Conference, ASME HTD-340, vol. 2, G. S. Dulikravich and K. A. Woodbury (editors), 109-116, June (1997).
[83] C. H. Huang and C. H. Tsai, A Shape Identification Problem in Estimating Time-Dependent Irregular Boundary Configurations, National Heat Transfer Conference, ASME HTD-340, vol. 2, G. S. Dulikravich and K. A. Woodbury (editors), 41-48 (1997).
[84] J. P. Alencar Jr., H. R. B. Orlande and M. N. Ozisik, A Generalized Coordinates Approach for the Solution of Inverse Heat Conduction Problems, 11th International Heat Transfer Conference, Korea, 7, 53-58, August (1998).

[85] M. J. Colaço and H. R. B. Orlande, A Natural Convection Inverse Problem of Simultaneous Identification of Two Boundary Heat Fluxes in Rectangular Cavities, In: 12th International Heat Transfer Conf., Grenoble, France (2002).
[86] M. J. Colaço and H. R. B. Orlande, Estimation of the Heat Transfer Coefficient at the Surface of a Plate by Using the Conjugate Gradient Method, In: VII Encontro Nacional de Ciências Térmicas, Rio de Janeiro, Brazil, 1, 189-194 (1998).
[87] M. J. Colaço and H. R. B. Orlande, A Function Estimation Approach for the Identification of the Transient Inlet Profile in Parallel Plate Channels, In: International Symposium on Inverse Problems in Engineering Mechanics, M. Tanaka and G. S. Dulikravich (editors), 409-418, Nagano City, Japan, (2000).
[88] C. Darwin, On the Origin of Species, John Murray, London (1859).
[89] D. Sahoo and G. S. Dulikravich, Evolutionary Wavelet Neural Network for Large Scale Function Estimation in Optimization, 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA Paper AIAA-2006-6955, Portsmouth, VA, September 6-8 (2006).
[90] E. J. Kansa, Multiquadrics – A Scattered Data Approximation Scheme with Applications to Computational Fluid Dynamics – II: Solutions to Parabolic, Hyperbolic and Elliptic Partial Differential Equations, Comput. Math. Applic., 19, 149-161 (1990).
[91] R. L. Hardy, Multiquadric Equations of Topography and Other Irregular Surfaces, Journal of Geophysics Res., 176, 1905-1915 (1971).
[92] V. M. A. Leitão, A Mesheless Method for Kirchhoff Plate Bending Problems, International Journal of Numerical Methods in Engineering, 52, 1107-1130, (2001).
[93] V. M. A. Leitão, RBF-Based Meshless Methods for 2D Elastostatic Problems, Engineering Analysis with Boundary Elements, 28, 1271-1281 (2004).
[94] H. Wendland, Error Estimates for Interpolation by Compactly Supported Radial Basis Functions of Minimal Degree, Journal of Approximation Theory, 93, 258-272 (1998).
[95] P. Lancaster and K. Salkauskas, Curve and Surface Fitting: An Introduction, Academic Press, Harcourt Brace Jovanovic, London-San Diego-New York (1986).
[96] M. Kaufman, V. Balabanov, S. L. Burgee, A. A. Giunta, B. Grossman, W. H. Mason and L. T. Watson, Variable Complexity Response Surface Approximations for Wing Structural Weight in HSCT Design, AIAA Paper 96-0089, Proc. 34th Aerospace Sciences Meeting and Exhibit, Reno, NV (1996).
[97] H. R. Madala and A. G. Ivakhnenko, Inductive Learning Algorithms for Complex Systems Modeling, CRC Press, Boca Raton, Florida, USA (1994).
[98] R. J. Moral and G. S. Dulikravich, A Hybrid Self-Organizing Response Surface Methodology, paper AIAA-2008-5891, 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, BC, Canada, September 10-12 (2008).
[99] IOSO NM Version 1.0, User's Guide, IOSO Technology Center, Moscow, Russia (2003).
[100] Buhmann, M.D., Radial Basis Functions on Grids and Beyond, International Workshop on Meshfree Methods, Lisbon (2003).
[101] R. Jin, W. Chen and T. W. Simpson, Comparative Studies of Metamodeling Techniques under Multiple Modeling Criteria, Proceedings of the 8th AIAA / USAF / NASA / ISSMO Multidisciplinary Analysis & Optimization Symposium, AIAA 2000-4801, Long Beach, CA, 6-8 Sept (2000).
[102] W. Hock and K. Schittkowski, Test Examples for Nonlinear Programming Codes, Lecture Notes in Economics and Mathematical Systems, 187, Berlin / Heidelberg / New York: Springer-Verlag (1981).
[103] K. Schittkowski, More Test Examples for Nonlinear Programming", Lecture Notes in Economics and Mathematical Systems, 282, Springer Verlag (1987).
[104] J. J. More, B. S. Gabow and K. E. Hillstrom, Testing Unconstrained Optimization Software, ACM Trans. Meth. Software, 17-41 (1981).
[105] E. Sandgren, The Utility of Nonlinear Programming Algorithms, Ph.D. thesis, Purdue University, IN (1977).
[106] A. Miele and J. W. Cantrell, Study on a Memory Gradient Method for the Minimization of Functions, J. Optim. Theory and Applications, 3 (6), 459-470 (1969).