10

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

Marcelo J. Colaço and George S. Dulikravich

CONTENTS

10.1 Introduction

Inverse problems usually involve the minimization of some objective function as part of their formulation. Such minimization procedures require the use of an optimization technique. Thus, in this chapter, we address solution methodologies for single-objective optimization problems, based on minimization techniques. Several gradient-based and non-gradient-based (stochastic) techniques are discussed, 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 (DFP) method (Levenberg, 1944; Hestenes and Stiefel, 1952; Davidon, 1959; Fletcher and Powell, 1963; Marquardt, 1963; Fletcher and Reeves, 1964; Broyden, 1965, 1967; Daniel, 1971; Polak, 1971; Beale, 1972; Bard, 1974; Beck and Arnold, 1977; Moré, 1977; Powell, 1977; Tikhonov and Arsenin, 1977; Dennis and Schnabel, 1983; Beck et al., 1985; Stoecker, 1989; Murio, 1993; Alifanov, 1994; Alifanov et al., 1995; Kurpisz and Nowak, 1995; Dulikravich and Martin, 1996; Trujillo and Busby, 1997; Jaluria, 1998; Beck, 1999; Belegundu and Chandrupatla, 1999; Colaço and Orlande, 1999, 2001a,b, 2004; Fletcher, 2000; Ozisik and Orlande, 2000; Woodbury, 2002). In addition, we present some of the stochastic approaches, such as the simulated annealing method (Corana et al., 1987; Goffe et al., 1994), the differential evolutionary method (Storn and Price, 1996), genetic algorithms (Goldberg, 1989; Deb, 2002), and the particle swarm method (Kennedy and Eberhart, 1995; Kennedy, 1999; Eberhart et al., 2001; Naka et al., 2001). 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 nonpractical. In order to overcome these difficulties, we will also discuss the so-called hybrid algorithms that take advantage of the robustness of the stochastic methods and the fast convergence of the deterministic methods (Dulikravich et al., 1999, 2003, 2004, 2008; Colaço and Orlande, 2001a,b; Colaço et al., 2004, 2005, 2006, 2008; Colaço and Dulikravich, 2006, 2007; Dulikravich and Colaço, 2006; Wellele et al., 2006; Silva et al., 2007; Padilha et al., 2009). 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 (Dulikravich et al., 1999; Colaço et al., 2005, 2008). AQ1

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 multicomponent alloy requires manufacturing each candidate alloy and evaluating its properties using classical experimental techniques. Even with the most efficient optimization algorithms (Dulikravich et al., 2008), this means that often thousands 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. Aerodynamics of thousands of different shapes needs to be analyzed using computational fluid dynamics software, which would be unacceptably time consuming.

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 or a response surface) instead of the actual high fidelity analysis method. Such surrogate models are often known as response surfaces (Colaço et al., 2007, 2008). In the 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. Therefore, we also present in this chapter some basic concepts related to the response surface generation methodology.

10.2 Basic Concepts

10.2.1 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

$$S = S(\mathbf{P}); \quad \mathbf{P} = \{P_1, P_2, \ldots, P_N\} \tag{10.1}$$

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

The relationship between S and $\mathbf{P}$ can, most of the time, 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 $\mathbf{P}$ must be determined by experiments.

10.2.2 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 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 10.1, 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.

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 10.1: 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 10.1.

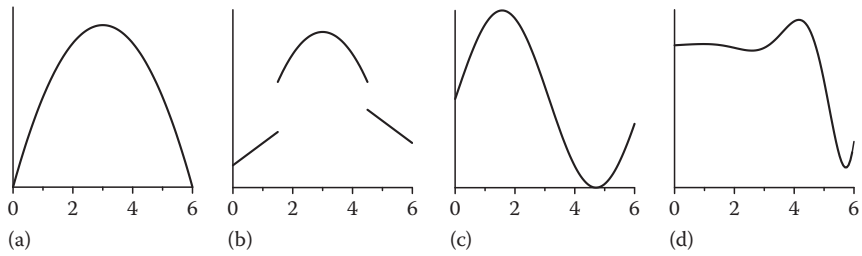

FIGURE 10.1
Some examples of functions S (ordinate) of a single design variable P (abscissa). AQ3

10.2.3 Single- and Multi-Objective Functions

This chapter will deal only with single-objective functions. However, it is interesting to introduce the reader to the multi-objective optimization problems (Deb, 2002) 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 (e.g., fuel consumption), 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 10.2.

10.2.4 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 prespecified 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

$$G = G(\mathbf{P}) = 0 \tag{10.2}$$

This kind of constraint can represent, for example, the prespecified power of an automobile.

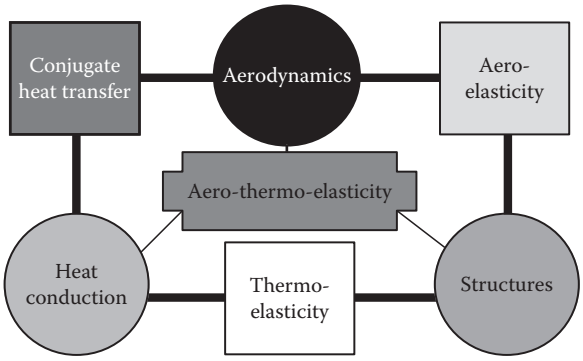

FIGURE 10.2
An example of a multi-objective design optimization problem.

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

$$Q = Q(\mathbf{P}) < 0 \tag{10.3}$$

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

10.2.5 Optimization Problems

Inverse problems are mathematically classified as *ill-posed*, whereas standard heat transfer problems are *well-posed*. The solution of a well-posed problem must satisfy the conditions of existence, uniqueness, and stability with respect to the input data (Hadamard, 1923). The existence of a solution for an inverse heat transfer problem may be assured by physical reasoning. On the other hand, the uniqueness of the solution of inverse problems can be mathematically proved only for some special cases. Also, the inverse problem is very sensitive to random errors in the measured input data, thus requiring special techniques for its solution in order to satisfy the stability condition.

Successful solution of an inverse problem generally involves its reformulation as an approximate well-posed problem and makes use of some kind of regularization (stabilization) technique. Although the solution techniques for inverse problems do not necessarily make use of optimization techniques, many popular methods are based on them.

Despite their similarities, inverse and optimization problems are conceptually different. *Inverse problems are concerned with the identification of unknown quantities appearing in the mathematical formulation of physical problems, by using measurements of the system response*. On the other hand, *optimization problems generally deal with the minimization or maximization of a certain objective or cost function, in order to find design variables that will result in extreme value of the objective function*. In addition, inverse and optimization problems involve other different concepts. For example, the solution technique for an inverse problem is required to cope with instabilities resulting from the noisy measured input data, while for an optimization problem, the input data is given by the desired response(s) of the system. In contrast to inverse problems, the solution uniqueness may not be an important issue for optimization problems, as long as the solution obtained is physically feasible and can be practically implemented. Engineering applications of optimization techniques are very often concerned with the minimization or maximization of different quantities, such as minimum weight (e.g., lighter airplanes), minimum fuel consumption (e.g., more economic cars), maximum autonomy (e.g., longer range airplanes), etc. The necessity of finding the maximum or minimum values of some parameters (or functions) can be governed by economic factors, as in the case of fuel consumption, or design characteristics, as in the case of maximum autonomy of an airplane. Sometimes, however, the decision is more subjective, as in the case of choosing a car model. In general, different designs can be idealized for a given application, but only a few of them will be economically viable.

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 chapter, 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 (Hadamard, 1923; Daniel, 1971; Beck and Arnold, 1977; Tikhonov and Arsenin, 1977; Sabatier, 1978; Morozov, 1984; Beck et al., 1985; Hensel, 1991; Murio, 1993; Alifanov, 1994; Alifanov et al., 1995; Kurpisz and Nowak, 1995; Dulikravich and Martin, 1996; Kirsch, 1996; Trujillo and Busby, 1997; Isakov, 1998; Beck, 1999; Denisov, 1999; Yagola et al., 1999; Zubelli, 1999; Ozisik and Orlande, 2000; Ramm et al., 2000; Woodbury, 2002).

10.3 Deterministic Methods

In this section, some deterministic methods like the steepest descent method, the conjugate gradient method, the Newton–Raphson, and the quasi-Newton methods will be discussed. Some practical aspects and limitations of such methods will be addressed.

These types of methods, as applied to nonlinear 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 (Bard, 1974; Beck and Arnold, 1977; Dennis and Schnabel, 1983; Stoecker, 1989; Alifanov, 1994; Alifanov et al., 1995; Jaluria, 1998; Belegundu and Chandrupatla, 1999; Fletcher, 2000):

$$\mathbf{P}^{k+1} = \mathbf{P}^k + \alpha^k \mathbf{d}^k \tag{10.4}$$

where

$\mathbf{P}$ is the vector of design variables
α is the search step size
$\mathbf{d}$ is the direction of descent
k is the iteration number

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 $\mathbf{d} = -\mathbf{R}\nabla S$ (Bard, 1974).

Such requirement results in directions of descent that form an angle greater than 90° with the gradient direction. A minimization method in which the directions are obtained in this manner is called an *acceptable gradient method* (Bard, 1974).

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 (Bard, 1974).

10.3.1 Steepest Descent Method

The most basic gradient-based method is the steepest descent method (Daniel, 1971; Stoecker, 1989; Jaluria, 1998; Belegundu et al., 1999). 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 10.3.

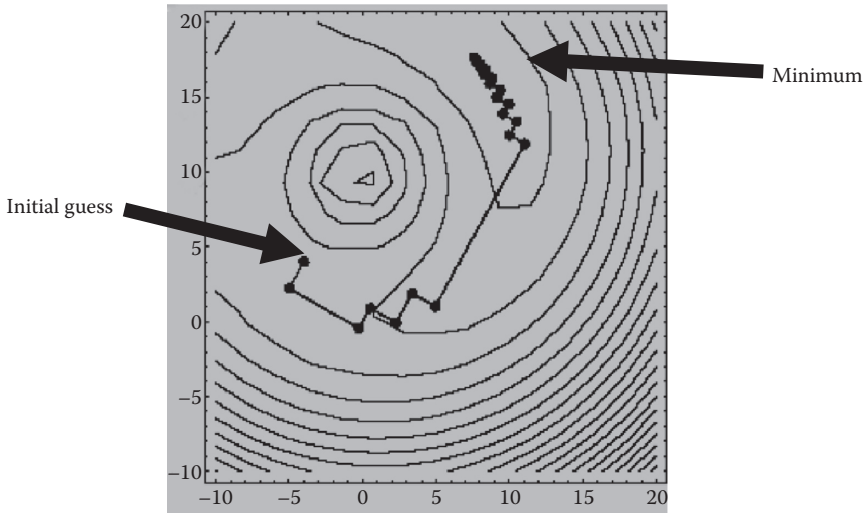

FIGURE 10.3
Convergence history for the steepest descent method.

The objective function can be mathematically stated as

$$S = S(\mathbf{P}); \quad \mathbf{P} = \{P_1, P_2, \ldots, P_N\} \tag{10.5}$$

The direction in which the objective function S varies most rapidly is the direction of gradient of S. For example, for the case with two variables (Figure 10.3), the gradient is

$$\nabla S = \frac{\partial S}{\partial P_1}\mathbf{i}_1 + \frac{\partial S}{\partial P_2}\mathbf{i}_2 \tag{10.6}$$

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

$$\mathbf{P}^{k+1} = \mathbf{P}^k - \alpha^k \nabla S(\mathbf{P}^k) \tag{10.7}$$

where
$\mathbf{P}$ is the vector of variables being optimized
α is the search step size
k is a counter for the iterations

Comparing Equations 10.4 and 10.7, one can check that for the steepest descent method, the direction of descent $\mathbf{d}$ is given by

$$\mathbf{d}^k = -\nabla S(\mathbf{P}^k) \tag{10.8}$$

In spite of this being the natural choice for the direction of descent, it is not very efficient as can be seen in Figure 10.3. 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 10.7 and the definition of the objective function (10.1), we have that at iteration level $k+1$,

$$S(\mathbf{P}^{k+1}) = S(\mathbf{P}^k + \alpha^k \mathbf{d}^k) \tag{10.9}$$

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

$$\frac{dS(\mathbf{P}^{k+1})}{d\alpha^k} = 0 \tag{10.10}$$

Using the chain rule,

$$\frac{dS(\mathbf{P}^{k+1})}{d\alpha^k} = \frac{dS(P_1^{k+1})}{dP_1^{k+1}} \frac{dP_1^{k+1}}{d\alpha^k} + \frac{dS(P_2^{k+1})}{dP_2^{k+1}} \frac{dP_2^{k+1}}{d\alpha^k} + \cdots + \frac{dS(P_N^{k+1})}{dP_N^{k+1}} \frac{dP_N^{k+1}}{d\alpha^k} \tag{10.11}$$

Or

$$\frac{dS(\mathbf{P}^{k+1})}{d\alpha^k} = \left\langle [\nabla S(\mathbf{P}^{k+1})]^T, \frac{d\mathbf{P}^{k+1}}{d\alpha^k} \right\rangle \tag{10.12}$$

However, from Equations 10.7 and 10.8, it follows that

$$\frac{d\mathbf{P}^{k+1}}{d\alpha^k} = \mathbf{d}^k = -\nabla S(\mathbf{P}^k) \tag{10.13}$$

Substituting Equation 10.13 into (10.12) and (10.10), it follows that for steepest descent (Figure 10.4)

$$\left\langle [\nabla S(\mathbf{P}^{k+1})]^T, \nabla S(\mathbf{P}^k) \right\rangle = 0 \tag{10.14}$$

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 10.3).

In ''real life'' applications, it is not possible to use Equation 10.14 to evaluate the search step size, α. 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 (Stoecker, 1989; Jaluria, 1998), Fibonacci search (Stoecker, 1989; Jaluria, 1998), golden search (Stoecker, 1989; Jaluria, 1998), and cubic spline interpolation (de Boor, 1978), 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.

10.3.1.1 Exhaustive Search

AQ4

This method (Stoecker, 1989; Jaluria, 1998) is one of the less efficient search methods available for sequential computation (which means not parallel computation). However,

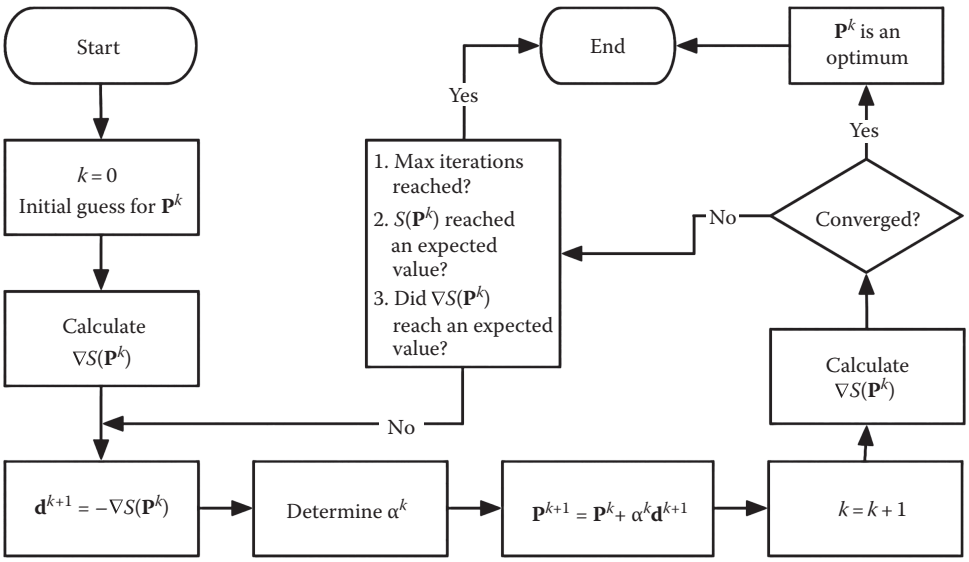

FIGURE 10.4
Iterative procedure for the steepest descent method.

it is a very good approach for parallel computing. 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 5 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 10.5, where an uncertainty interval I_0 was divided into eight subregions, which are not necessarily the same size.

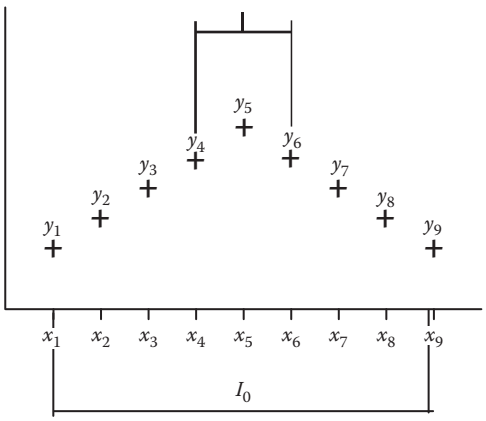

FIGURE 10.5
Exhaustive search method.

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

$$\begin{aligned} y_1 < y_2 < y_3 < y_4 < y_5 \\ y_5 > y_6 > y_7 > y_8 > y_9 \end{aligned} \tag{10.15}$$

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.

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 (Stoecker, 1989; Jaluria, 1998) that I is given by

$$I = \frac{2I_0}{n+1} \tag{10.16}$$

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 10.5, 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 five 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 10.5 is presented in Figure 10.6 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 10.5.

10.3.1.2 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 (Dulikravich and Martin, 1994), B-splines (de Boor, 1978), kriging (Oliver and Webster, 1990), or other interpolants. Interrogating these interpolated values, we can find

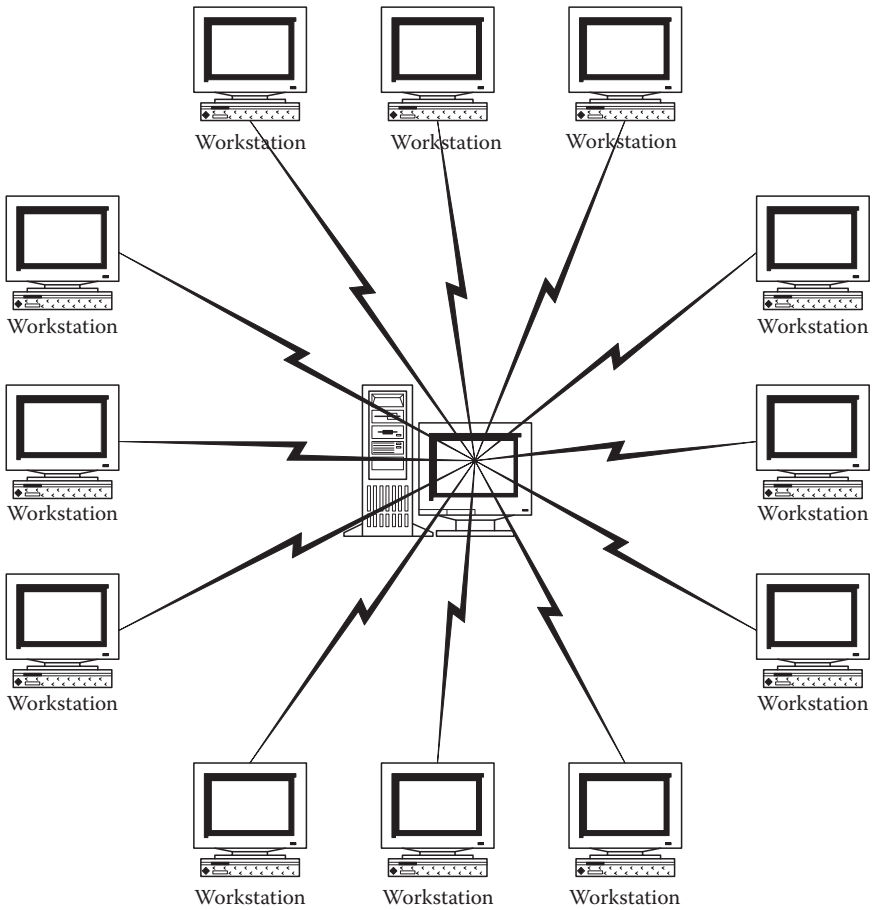

FIGURE 10.6
Typical setup for a parallel computing.

the region where the maximum or minimum values are located. The process can be repeated until a sufficiently small interval of uncertainty is obtained.

10.3.2 Conjugate Gradient Method

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 (Hestenes and Stiefel, 1952; Fletcher and Reeves, 1964; Daniel, 1971; Polak, 1971; Beale, 1972; Alifanov, 1974, 1994; Powell, 1977; Stoecker, 1989; Jarny et al., 1991; Artyukhin, 1993; Truffart et al., 1993; Dantas and Orlande, 1996; Huang and Tsai, 1997; Machado and Orlande, 1997; Orlande et al., 1997; Alencar Jr. et al., 1998; Colaço and Orlande, 1998; Jaluria, 1998; Belegundu and Chandrupatla, 1999; Colaço and Orlande, 2000, 2001a,b, 2002) 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 general equation used in the steepest descent method, Equation 10.4. 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, given as

$$\mathbf{d}^{k+1} = -\nabla S(\mathbf{P}^k) + \gamma^k \mathbf{d}^{k-1} + \psi^k \mathbf{d}^q \tag{10.17}$$

where γ^k and ψ^k are conjugation coefficients. The superscript q in Equation 10.17 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 (1977) 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 10.17 (Hestenes and Stiefel, 1952; Fletcher and Reeves, 1964; Daniel, 1971; Polak, 1971; Beale, 1972; Alifanov, 1974, 1994; Powell, 1977; Jarny et al., 1991; Artyukhin, 1993; Truffart et al., 1993; Dantas and Orlande, 1996; Machado and Orlande, 1997; Orlande et al., 1997; Alencar Jr. et al., 1998). In the Fletcher–Reeves version (Fletcher and Reeves, 1964), the conjugation coefficients γ^k and ψ^k are obtained from the following expressions (Fletcher and Reeves, 1964; Daniel, 1971; Alifanov, 1974, 1994; Powell, 1977; Jarny et al., 1991; Dantas and Orlande, 1996; Huang and Tsai, 1997; Machado and Orlande, 1997; Orlande et al., 1997):

$$\gamma^k = \frac{\| \nabla S(\mathbf{P}^k) \|^2}{\| \nabla S(\mathbf{P}^{k-1}) \|^2}, \quad \text{with } \gamma^0 = 0 \quad \text{for } k = 0 \tag{10.18a}$$

$$\psi^k = 0, \quad \text{for } k = 0, 1, 2 \tag{10.18b}$$

In the Polak–Ribiere version of the conjugate gradient method (Daniel, 1971; Polak, 1971; Powell, 1977; Jarny et al., 1991; Artyukhin, 1993; Truffart et al., 1993; Alifanov, 1994), the conjugation coefficients are given by

$$\gamma^k = \frac{[\nabla S(\mathbf{P}^k)]^T [\nabla S(\mathbf{P}^k) - \nabla S(\mathbf{P}^{k-1})]}{\| \nabla S(\mathbf{P}^{k-1}) \|^2}, \quad \text{with } \gamma^0 = 0 \quad \text{for } k = 0 \tag{10.19a}$$

$$\psi^k = 0, \quad \text{for } k = 0, 1, 2, \ldots \tag{10.19b}$$

Based on a previous work by Beale (1972), Powell (1977) suggested the following expressions for the conjugation coefficients, which gives the so-called Powell–Beale's version of the conjugate gradient method (Beale, 1972; Alifanov, 1974; Powell, 1977):

$$\gamma^k = \frac{[\nabla S(\mathbf{P}^k)]^T [\nabla S(\mathbf{P}^k) - \nabla S(\mathbf{P}^{k-1})]}{[\mathbf{d}^{k-1}]^T [\nabla S(\mathbf{P}^k) - \nabla S(\mathbf{P}^{k-1})]}, \quad \text{with } \gamma^0 = 0 \quad \text{for } k = 0 \tag{10.20a}$$

$$\psi^k = \frac{[\nabla S(\mathbf{P}^k)]^T [\nabla S(\mathbf{P}^{q+1}) - \nabla S(\mathbf{P}^q)]}{[\mathbf{d}^q]^T [\nabla S(\mathbf{P}^{q+1}) - \nabla S(\mathbf{P}^q)]}, \quad \text{with } \gamma^0 = 0 \quad \text{for } k = 0 \tag{10.20b}$$

In accordance with Powell (1977), the application of the conjugate gradient method with the conjugation coefficients given by Equations 10.20 requires restarting when gradients at successive iterations tend to be non-orthogonal (which is a measure of the local nonlinearity of the problem) and when the direction of descent is not sufficiently downhill. Restarting is performed by making $\psi^k = 0$ in Equation 10.17.

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

$$\text{ABS}([\nabla S(\mathbf{P}^{k-1})]^T \nabla S(\mathbf{P}^k)) \geq 0.2 \, \| \nabla S(\mathbf{P}^k) \|^2 \tag{10.21a}$$

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:

$$[\mathbf{d}^k]^T \nabla S(\mathbf{P}^k) \leq -1.2 \, \| \nabla S(\mathbf{P}^k) \|^2 \tag{10.21b}$$

$$[\mathbf{d}^k]^T \nabla S(\mathbf{P}^k) \geq -0.8 \, \| \nabla S(\mathbf{P}^k) \|^2 \tag{10.21c}$$

We note that the coefficients 0.2, 1.2, and 0.8 appearing in Equations 10.21a through 10.21c are empirically determined and are the same values used by Powell (1977).

In Powell–Beale's version of the conjugate gradient method, the direction of descent given by Equation 10.17 is computed in accordance with the following algorithm for $k \geq 1$ (Powell, 1977):

Step 1: Test the inequality (10.21a). If it is true, set $q = k - 1$.

Step 2: Compute γ^k using Equation 10.20a.

Step 3: If $k = q + 1$, set $\psi^k = 0$. If $k \neq q + 1$, compute ψ^k using Equation 10.20b.

Step 4: Compute the search direction $\mathbf{d}^{k+1}$ using Equation 10.17.

Step 5: If $k \neq q + 1$, test the inequalities (10.21b and 10.21c). If either one of them is satisfied, set $q = k - 1$ and $\psi^k = 0$. Then, recompute the search direction using Equation 10.17.

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 10.17. We note that the conjugation coefficients γ^k given by Equations 10.18a, 10.19a, and 10.20a are equivalent for quadratic functions, because the gradients at different iterations are mutually orthogonal (Daniel, 1971; Powell, 1977).

The same procedures used for the evaluation of the search step size in the steepest descent method can be employed here. Figure 10.7 illustrates the convergence history for the Fletcher–Reeves version of the conjugate gradient method for the same function presented in Figure 10.3. 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.

Colaço and Orlande (1999) 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

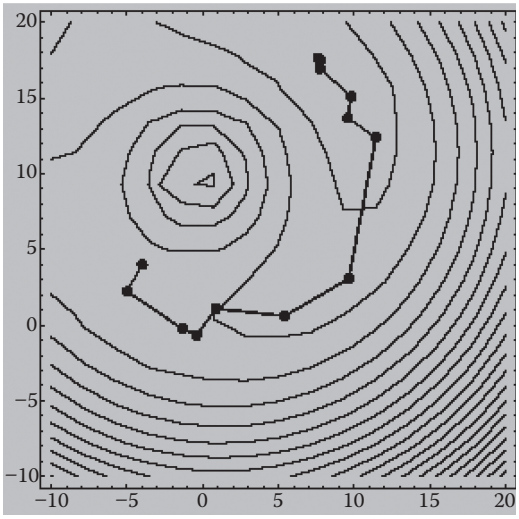

FIGURE 10.7
Convergence history for the Fletcher–Reeves version of the conjugate gradient method.

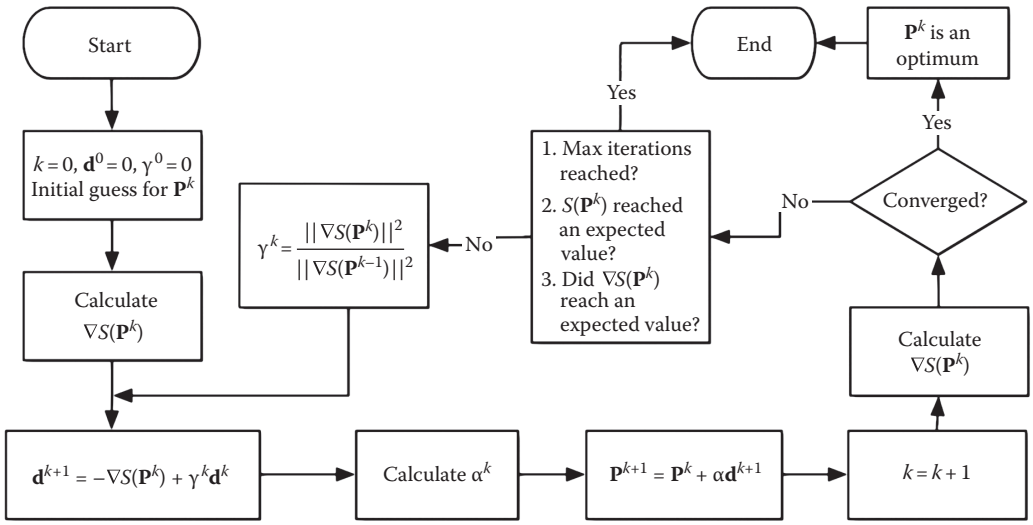

FIGURE 10.8
Iterative procedure for the Fletcher–Reeves version of the conjugate gradient method.

the RMS error of the estimated functions were obtained with Powell–Beale's version of the conjugate gradient method. AQ5

Figure 10.8 shows the iterative procedure for the Fletcher–Reeves version (Fletcher and Reeves, 1964) of the conjugate gradient method.

10.3.3 Newton–Raphson Method

While the steepest descent and the conjugate gradient methods use gradients of the objective function in their iterative procedures, the Newton–Raphson method (Daniel, 1971; Stoecker, 1989; Jaluria, 1998; Belegundu and Chandrupatla, 1999) 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

$$S(\mathbf{P}+\mathbf{h}) = S(\mathbf{P}) + \nabla S(\mathbf{P})^T \mathbf{h} + \frac{1}{2}\mathbf{h}^T D^2 S(\mathbf{P})\mathbf{h} + O(\mathbf{h}^3) \tag{10.22}$$

where $\nabla S(\mathbf{P})$ is the gradient (vector of first-order derivatives), while $D^2S(\mathbf{P})$ is the Hessian (matrix of second-order derivatives).

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

$$\nabla S(\mathbf{P}+\mathbf{h}) \cong \nabla S(\mathbf{P}) + D^2 S(\mathbf{P})\mathbf{h} \tag{10.23}$$

The optimum is obtained when the left side of Equation 10.23 vanishes. Thus, we have

$$\mathbf{h}_{\text{optimum}} \cong -[D^2 S(\mathbf{P})]^{-1} \nabla S(\mathbf{P}) \tag{10.24}$$

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

$$(\mathbf{P} + \mathbf{h}_{\text{optimum}}) \cong \mathbf{P} - [D^2 S(\mathbf{P})]^{-1} \nabla S(\mathbf{P}) \tag{10.25}$$

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 10.4 where the direction of descent is given by

$$\mathbf{d}^{k+1} = -[D^2 S(\mathbf{P}^k)]^{-1} \nabla S(\mathbf{P}^k) \tag{10.26}$$

The Newton–Raphson method is faster than the conjugate gradient method as demonstrated in Figure 10.9. However, the calculation of the Hessian matrix coefficients takes a long time. Figure 10.10 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.

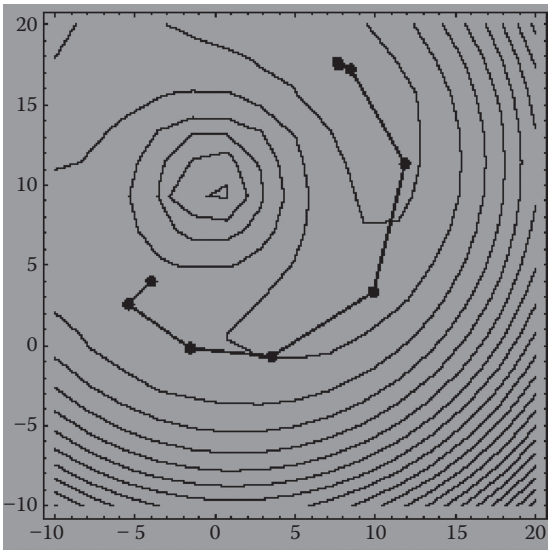

FIGURE 10.9
Convergence history for the Newton–Raphson method.

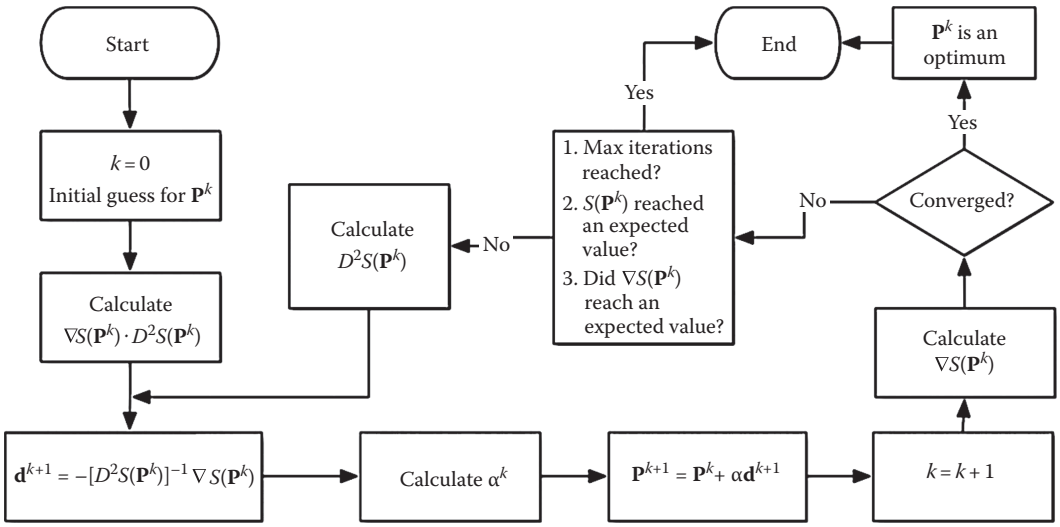

FIGURE 10.10
Iterative procedure for the basic Newton–Raphson method implementation.

10.3.4 Quasi-Newton Methods

The quasi-Newton methods (Daniel, 1971; Stoecker, 1989; Jaluria, 1998; Belegundu and Chandrupatla, 1999) 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

$$\mathbf{H}^k = [D^2S(\mathbf{P}^k)]^{-1} \tag{10.27}$$

Thus, the quasi-Newton methods follow the general iterative procedure given by Equation 10.4, where the direction of descent is given by

$$\mathbf{d}^{k+1} = -\mathbf{H}^k \nabla S(\mathbf{P}^k) \tag{10.28}$$

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

$$\mathbf{H}^k = \mathbf{H}^{k-1} + \mathbf{M}^{k-1} + \mathbf{N}^{k-1} \quad \text{for } k = 1, 2, \ldots \tag{10.29a}$$

$$\mathbf{H}^k = \mathbf{I} \quad \text{for } k = 0 \tag{10.29b}$$

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 DFP method (Davidon, 1959; Fletcher and Powell, 1963), such matrices are given by

$$\mathbf{M}^{k-1} = \alpha^{k-1} \frac{\mathbf{d}^{k-1}(\mathbf{d}^{k-1})^T}{(\mathbf{d}^{k-1})^T \mathbf{Y}^{k-1}} \tag{10.30a}$$

$$\mathbf{N}^{k-1} = -\frac{(\mathbf{H}^{k-1}\mathbf{Y}^{k-1})(\mathbf{H}^{k-1}\mathbf{Y}^{k-1})^T}{(\mathbf{Y}^{k-1})^T \mathbf{H}^{k-1}\mathbf{Y}^{k-1}} \tag{10.30b}$$

where

$$\mathbf{Y}^{k-1} = \nabla S(\mathbf{P}^k) - \nabla S(\mathbf{P}^{k-1}) \tag{10.30c}$$

Figure 10.11 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.

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 (Colaço et al., 2006). 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 (Davidon, 1959; Fletcher and Powell, 1963; Broyden, 1965, 1967), which is less sensitive to the choice of the search step size. For this method, the matrices **M** and **N** are calculated as

$$\mathbf{M}^{k-1} = \left(\frac{1 + (\mathbf{Y}^{k-1})^T \mathbf{H}^{k-1}\mathbf{Y}^{k-1}}{(\mathbf{Y}^{k-1})^T \mathbf{d}^{k-1}}\right) \frac{\mathbf{d}^{k-1}(\mathbf{d}^{k-1})^T}{(\mathbf{d}^{k-1})^T \mathbf{Y}^{k-1}} \tag{10.31a}$$

$$\mathbf{N}^{k-1} = -\frac{\mathbf{d}^{k-1}(\mathbf{Y}^{k-1})^T \mathbf{H}^{k-1} + \mathbf{H}^{k-1}\mathbf{Y}^{k-1}(\mathbf{d}^{k-1})^T}{(\mathbf{Y}^{k-1})^T \mathbf{d}^{k-1}} \tag{10.31b}$$

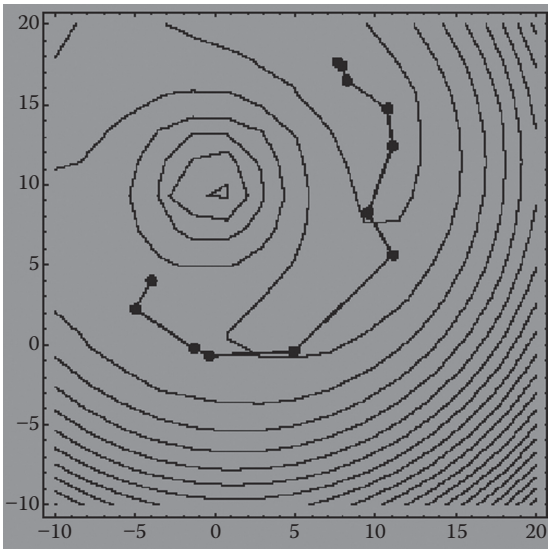

FIGURE 10.11
Convergence history for the DFP method.

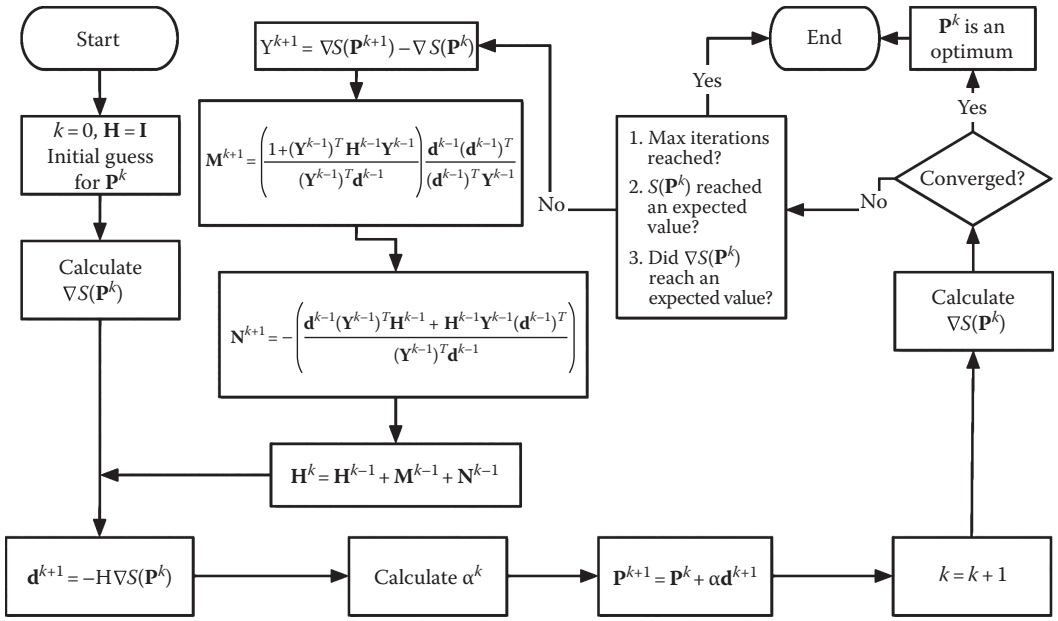

FIGURE 10.12
Iterative procedure for the BFGS method.

Figure 10.12 shows the 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 10.25. The search step size prevents the method from jumping to a maximum value when we look for a minimum and vice versa. Figures 10.13 and 10.14 show the influence of the initial guess for all four methods for a Rosenbrock ''banana-shape'' function (More et al., 1981).

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.

10.3.5 Levenberg–Marquardt Method

The Levenberg–Marquardt method was first derived by Levenberg (1944), by modifying the ordinary least squares norm. Later, in 1963, Marquardt (1963) 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

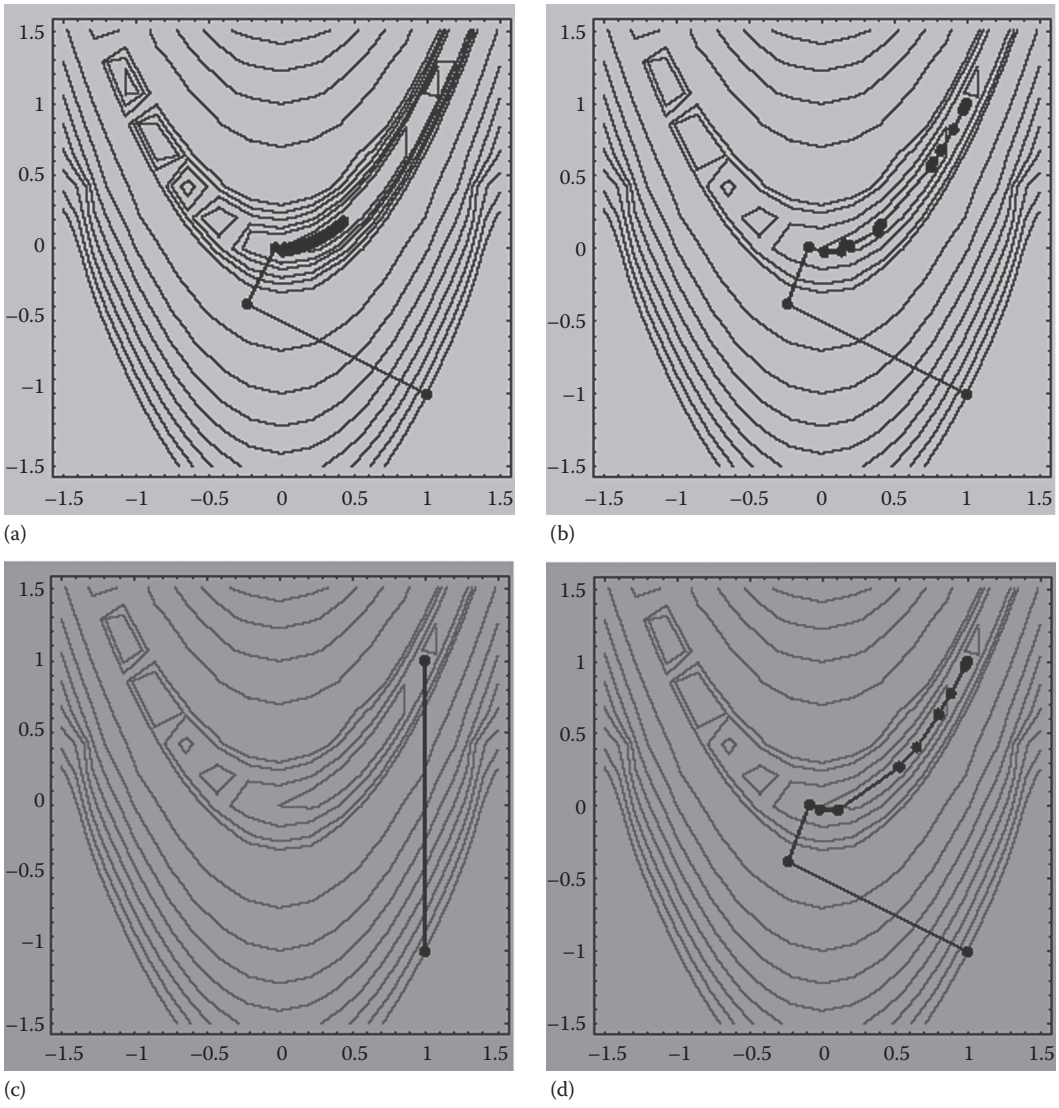

FIGURE 10.13
First initial guess for the (a) steepest descent, (b) conjugate gradient, (c) Newton–Raphson, and (d) DFP methods.

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 $\mathbf{J}$ is a positive definite matrix, then $\mathbf{A} + \lambda\mathbf{J}$ is positive definite for sufficiently large λ. If $\mathbf{A}$ is an approximation for the Hessian, we can

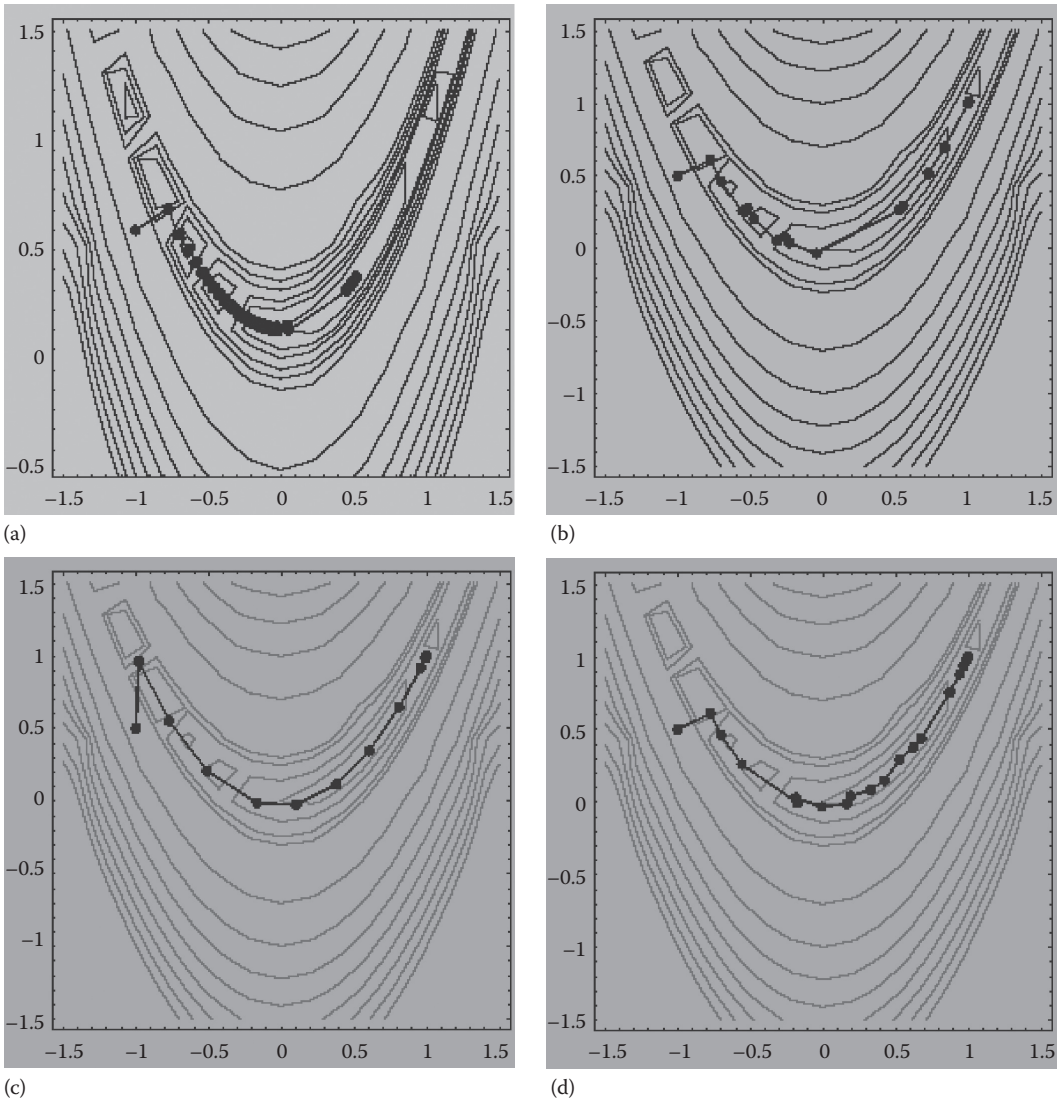

FIGURE 10.14
Second initial guess for the (a) steepest descent, (b) conjugate gradient, (c) Newton–Raphson, and (d) DFP methods.

choose **J** as a diagonal matrix whose elements coincide with the absolute values of the diagonal elements of **A** (Bard, 1974).

The direction of descent for the Levenberg–Marquardt method is given by (Bard, 1974)

$$\mathbf{d}^k = -(\mathbf{A}^k + \lambda^k \mathbf{J}^k)^{-1} \nabla S(\mathbf{P}^k) \tag{10.32}$$

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 (Bard, 1974; Beck and Arnold, 1977; Ozisik and Orlande, 2000).

10.4 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.

10.4.1 Genetic Algorithms

Genetic algorithms (Goldberg, 1989) 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 (Goldberg, 1989; Deb, 2002).

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 (e.g., 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 (Goldberg, 1989; Deb, 2002).

10.4.2 Differential Evolution

The differential evolution method (Storn and Price, 1996) is an evolutionary method based on Darwin's theory of evolution of the species (Darwin, 1859). This non-gradient-based optimization method was created in 1995 (Storn and Price, 1996) 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 10.15)

$$\mathbf{P}_i^{k+1} = \delta_1 \mathbf{P}_i^k + \delta_2[\boldsymbol{\alpha} + F(\boldsymbol{\beta} - \boldsymbol{\gamma})] \tag{10.33}$$

where

$\mathbf{P}_i$ is the ith 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 constant, which defines the mutation ($0.5 < F < 1$)
k is a counter for the generations
δ_1 and δ_2 are two 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.

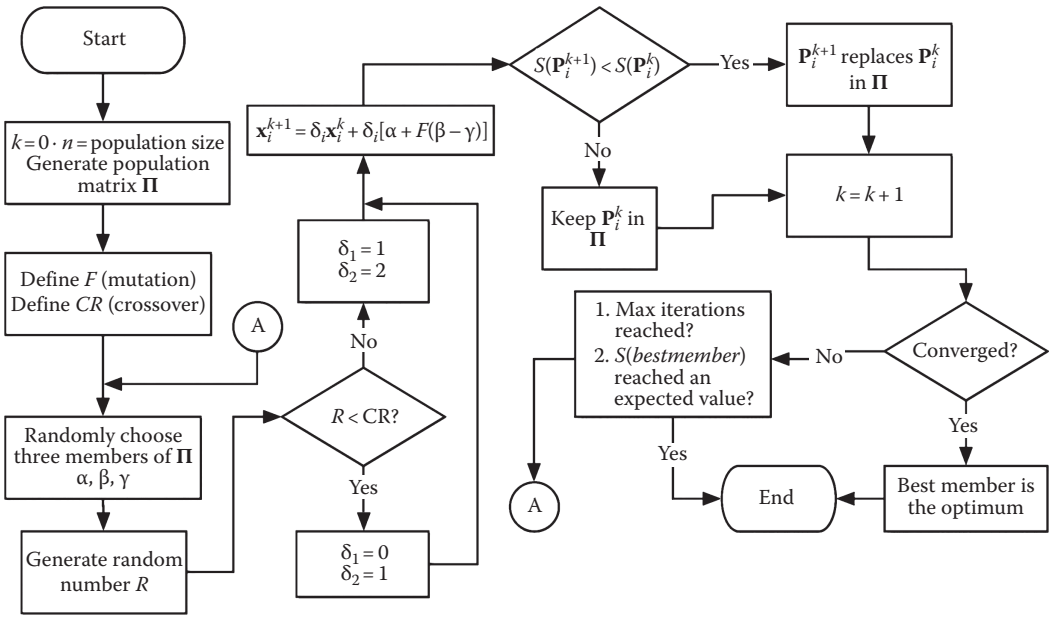

FIGURE 10.15
Iterative procedure for the differential evolution method.

The binomial crossover is given as

$$\delta_1 = \begin{cases} 0, & \text{if } R < CR \\ 1, & \text{if } R > CR \end{cases} \tag{10.34a}$$

$$\delta_2 = \begin{cases} 1, & \text{if } R < CR \\ 0, & \text{if } R > CR \end{cases} \tag{10.34b}$$

where

CR is a factor that defines the crossover ($0.5 < CR < 1$)

R is a random number with uniform distribution between 0 and 1

10.4.3 Particle Swarm

This non-gradient-based optimization method was created in 1995 by an electrical engineer (Russel Eberhart) and a social psychologist (James Kennedy) (Kennedy and Eberhart, 1995; Kennedy, 1999; Eberhart and Kennedy, 2001; Naka et al., 2001) 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

$$\mathbf{P}_i^{k+1} = \mathbf{P}_i^k + \mathbf{v}_i^{k+1} \tag{10.35a}$$

$$\mathbf{v}_i^{k+1} = \alpha \mathbf{v}_i^k + \beta \mathbf{r}_{1i}\left(\boldsymbol{\pi}_i - \mathbf{P}_i^k\right) + \beta \mathbf{r}_{2i}\left(\boldsymbol{\pi}_g - \mathbf{P}_i^k\right) \tag{10.35b}$$

where

$\mathbf{P}_i$ is the ith 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 ith individual, $\mathbf{P}_i$
$\boldsymbol{\pi}_g$ is the best value found by the entire population
$0 < \alpha < 1$; $1 < \beta < 2$

In Equation 10.35b, 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 ith component vector of parameters $\mathbf{P}_i$ during the iterative process. Thus, the individuality term involves the comparison between the current value of the ith 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 ith individual). Thus, the sociability term compares $\mathbf{P}_i$ with the best value of the entire population in the past.

Figure 10.16 shows the iterative procedure for the particle swarm method.

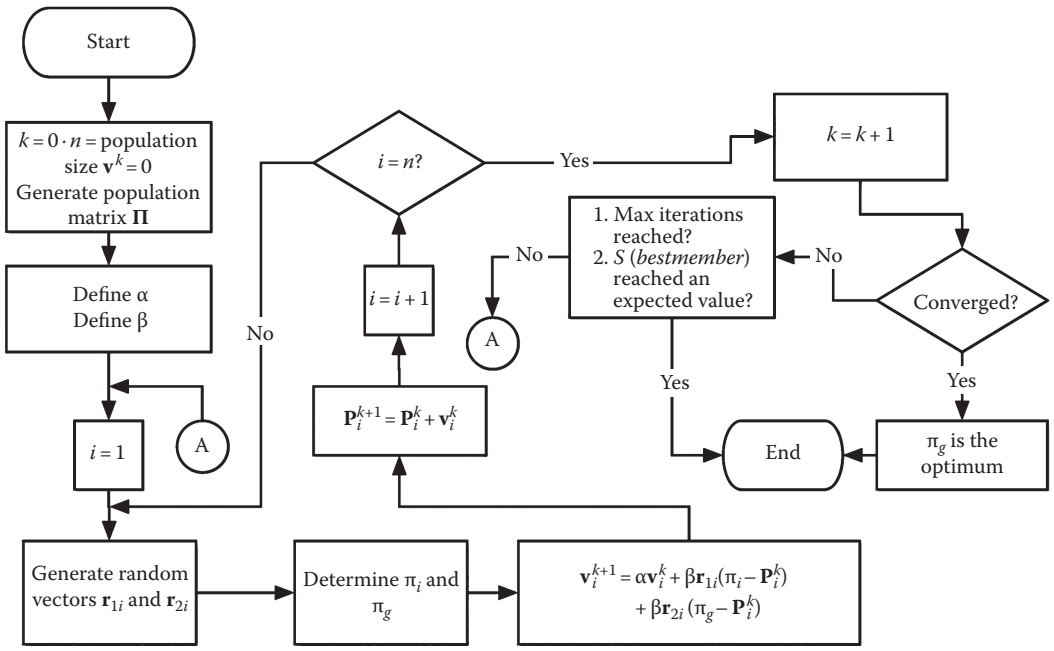

FIGURE 10.16
Iterative procedure for the particle swarm method.

10.4.4 Simulated Annealing

The simulated annealing method (Goffet et al., 1944; Corana et al., 1987) 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. AQ7

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

$$\text{Prob}(E) \propto e^{(-E/KT)} \tag{10.36}$$

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 10.36 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 10.17. 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 10.18 shows the iterative procedure for the simulated annealing method. The procedure starts generating a population of individuals of the same size as 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 $N_{it}^* N_s^* n$

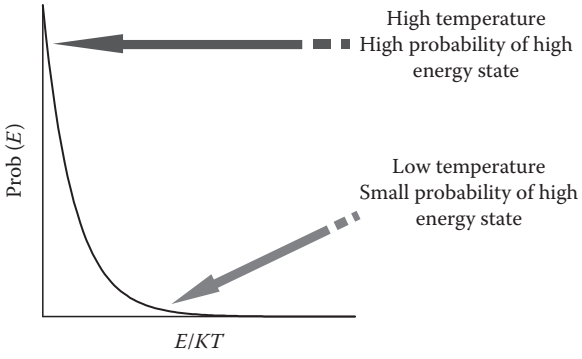

FIGURE 10.17
Schematic representation of Equation 10.36.

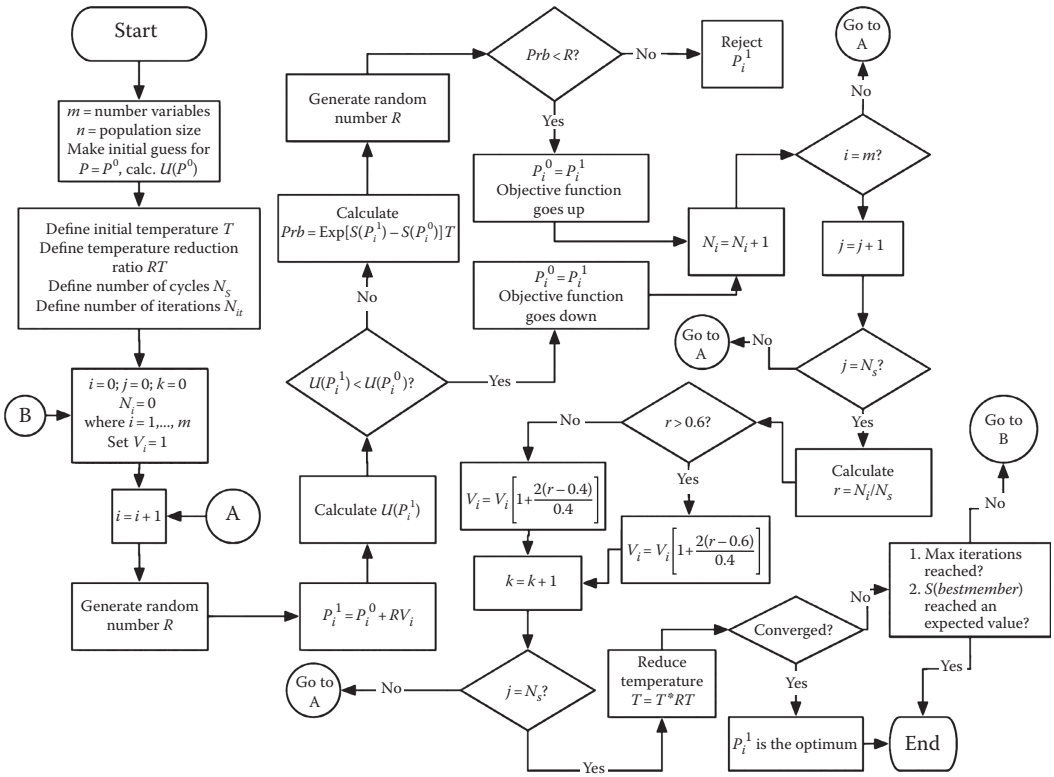

FIGURE 10.18
Iterative procedure for the simulated annealing method.

function evaluations, the temperature (T) is changed by the factor RT. The value suggested for the number of iterations by Corana et al. (1987) is MAX(100, 5*n).

The iterative process follows the equation

$$P_i^1 = P_i^0 + RV_i \tag{10.37}$$

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

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 $\left(P_i^1\right)$ and its value is compared to its value at the initial point $\left(P_i^0\right)$. 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 (Corana et al., 1987) criteria. It uses T (temperature) and the size of the downhill move in a probabilistic manner

$$P = e^{\left[S\left(P_i^1\right) - S\left(P_i^0\right)\right]/T} \tag{10.38}$$

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

$$T(i+1) = RT^*T(i) \tag{10.39}$$

where i is the ith 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, which produces a large enough V.

10.5 Hybrid Optimization Methods

The hybrid optimization methods (Dulikravich et al., 1999, 2003, 2004, 2008; Colaço and Orlande, 2001a,b; Colaço et al., 2004, 2005, 2006, 2008; Colaço and Dulikravich, 2006, 2007; Dulikravich and Colaço, 2006; Wellele et al., 2006; Silva et al., 2007; Padilha et al., 2009) 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 (Dulikravich et al., 1999).

One of the possible hybrid optimization methods encountered in the literature (Dulikravich et al., 1999, 2003, 2004, 2008; Colaço and Orlande, 2001a,b; Colaço et al., 2004, 2005, 2006, 2008; Colaço and Dulikravich, 2006, 2007; Dulikravich and Colaço, 2006; Wellele et al., 2006; Silva et al., 2007; Padilha et al., 2009), called in this chapter ***H1***, is illustrated in Figure 10.19 (Colaço et al., 2005). The driven module is very often the particle swarm method, which performs most of the optimization task. When a 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 10.16, the algorithm returns to the particle swarm method in order to AQ8
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.

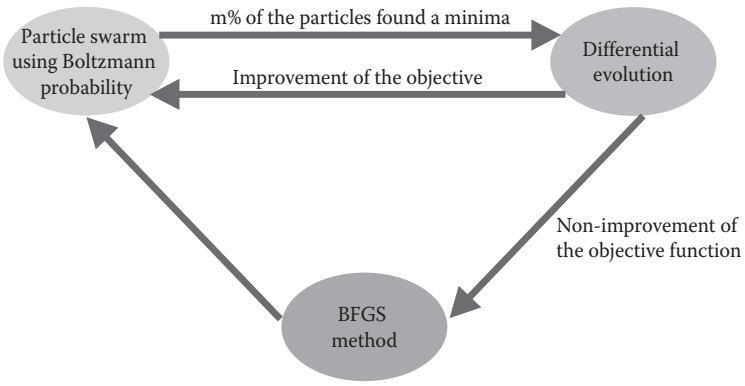

FIGURE 10.19
Global procedure for the hybrid optimization method ***H1***.

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.

10.6 Response Surfaces

From the viewpoint of kernel interpolation/approximation techniques, many response surface methods are based on linear and nonlinear 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 (Colaço et al., 2007) of an RBF-based response surface method and a wavelet-based artificial neural network method (Sahoo and Dulikravich, 2006) 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 (1990), after the work of Hardy (1991) 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 (2001, 2004).

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 (Wendland, 1998).

There are several other methods for automatically constructing multidimensional response surfaces. Notably, a classical book by Lancaster and Salkauskas (1986) offers a variety of methods for fitting hypersurfaces of a relatively small dimensionality. Kauffman et al. (1996) obtained reasonably accurate fits of data by using second-order polynomials. Ivakhnenko and his team in Ukraine (Madala and Ivakhnenko, 1994) have published an exceptionally robust method for fitting non-smooth data points in multidimensional spaces. Their method is based on a self-assembly approach where the analytical description of a hypersurface is a multilevel graph of the type ''polynomial-of-a-polynomial-of-a-polynomial-of-a-...'' and the basis functions are very simple polynomials (Moral and Dulikravich, 2008). This approach has been used in indirect optimization based upon self-organization (IOSO) (IOSO, 2003) commercial optimization software that has been known for its extraordinary speed and robustness.

10.6.1 RBF Model Used in This Chapter

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:

$$S(\mathbf{P}) \cong \xi(\mathbf{P}) = \sum_{j=1}^{N} \alpha_j \phi(|\mathbf{P} - \mathbf{P}_j|) + \sum_{k=1}^{M} \sum_{i=1}^{L} \beta_{i,k} q_k(P_i) + \beta_0 \tag{10.40}$$

where
$\mathbf{P} = P_1, \ldots, P_i, \ldots, P_L)$
$S(\mathbf{P})$ 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 (Buhmann, 2003). This approximation $\xi(\mathbf{P})$ is solved for the α_j and $\beta_{i,k}$ unknowns from the system of N linear equations, subject AQ9

$$\sum_{j=1}^{N} \alpha_j q_k(P_1) = 0$$

$$\vdots \tag{10.41}$$

$$\sum_{j=1}^{N} \alpha_j q_k(P_L) = 0$$

$$\sum_{j=1}^{N} \alpha_j = 0 \tag{10.42}$$

In this chapter, the polynomial part of Equation 10.40 was taken as

$$q_k(P_i) = P_i^k \tag{10.43}$$

and the RBFs are selected among the following:

$$\text{Multiquadrics:} \quad \phi(|P_i - P_j|) = \sqrt{(P_i - P_j)^2 + c_j^2} \tag{10.44a}$$

$$\text{Gaussian:} \quad \phi(|P_i - P_j|) = \exp\left[-c_j^2 (P_i - P_j)^2\right] \tag{10.44b}$$

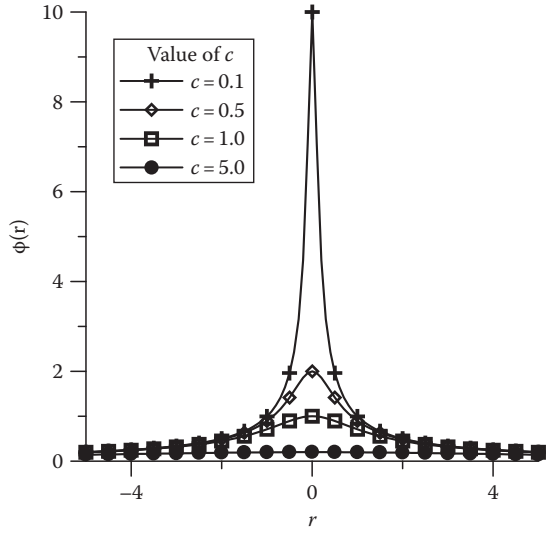

FIGURE 10.20
Influence of the shape parameter.

$$\text{Squared multiquadrics:} \quad \phi(|P_i - P_j|) = (P_i - P_j)^2 + c_j^2 \tag{10.44c}$$

$$\text{Cubical multiquadrics:} \quad \phi(|P_i - P_j|) = \left[\sqrt{(P_i - P_j)^2 + c_j^2}\right]^3 \tag{10.44d}$$

with the shape parameter c_j kept constant as $1/N$. The shape parameter is used to control the smoothness of the RBF. Figure 10.20 shows the influence on its choice for the multiquadrics RBF. From Equation 10.40, one can notice that a polynomial of order M is added to the RBF. M was limited to an upper value of 6. After inspecting Equations 10.40 through 10.43, 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.

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 suppose 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}. Equations 10.40 through 10.42 are solved for a polynomial of order zero and for the RBF expression given by Equations 10.44 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

$$RMS_{N_{TR1}, M=0, RBF_1} = \sum_{i=1}^{N_{TR2}} [S(P_i) - \xi(P_i)]^2 \tag{10.45}$$

Then, the same procedure is made, using the subset N_{TR2} to solve Equations 10.40 through 10.42 and the subset N_{TR1} to calculate the error as

$$RMS_{N_{TR2}, M=0, RBF_1} = \sum_{i=1}^{N_{TR1}} [S(P_i) - \xi(P_i)]^2 \tag{10.46}$$

Finally, the total error for the polynomial of order zero and the RBF expression given by Equations 10.44 is obtained as

$$RMS_{M=0,RBF_1} = \sqrt{RMS_{N_{TR1},M=0,RBF_1} + RMS_{N_{TR2},M=0,RBF_1}} \tag{10.47}$$

This procedure is repeated for all polynomial orders, up to $M = 6$ and for each one of the RBF expressions given by Equations 10.44. 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 analyzed in this chapter.

10.6.2 Performance Measurements

In accordance with having multiple metamodeling criteria, the performance of each metamodeling technique is measured from the following aspects (Jin et al., 2000):

- 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 (R^2), relative average absolute error ($RAAE$), and relative maximum absolute error ($RMAE$) (Jin et al., 2000).

10.6.2.1 R *Square*

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} = 1 - \frac{MSE}{\text{variance}} \tag{10.48}$$

where
$\hat{y}_i$ is the corresponding predicted value for the observed value y_i
$\bar{y}$ is the mean of the observed values

While mean square error (MSE) represents the departure of the metamodel from the real simulation model, the variance captures how irregular the problem is. The larger the value of R^2, the more accurate the metamodel.

10.6.2.2 Relative Average Absolute Error

$$RAAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i \|}{n^*STD} \tag{10.49}$$

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

10.6.2.3 Relative Maximum Absolute Error

$$RMAE = \frac{\max(|y_1 - \hat{y}_1|, |y_2 - \hat{y}_2|, \ldots, |y_n - \hat{y}_n|)}{STD} \tag{10.50}$$

Large *RMAE* indicates large error in one region of the design space even though the overall accuracy indicated by R^2 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 R^2 and *RAAE*.

Although the R^2, *RAAE*, and *RMAE* are useful to ascertain the accuracy of the interpolation, they can fail in some cases. For the R^2 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 10.48 and, even if all the other testing points are accurately interpolated. Similarly, the R^2 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 six ranges of errors: 0%–10%; 10%–20%; 20%–50%; 50%–100%; 100%–200%; >200%. Thus, an interpolation that has all testing points within the interval of 0%–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%.

10.6.3 Response Surface Test Cases

In order to show the accuracy of the RBF model presented, 296 test cases were used, representing linear and nonlinear 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 (1981) and Schittkowski (1987). Figure 10.21 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 10.40 through 10.42: 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.

In order to verify the accuracy of the interpolation over a different number of training points, three sets were defined. Also, the number of testing points varied, according to the number of training points. Table 10.1 presents these three sets, based on the number of dimensions (variables) L of the problem.

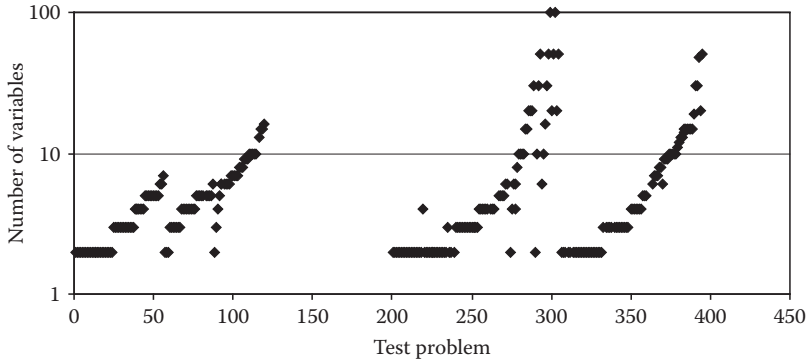

FIGURE 10.21
Number of variables for each problem considered.

TABLE 10.1
Number of Training and Testing Points

	Number of Training Points	Number of Testing Points
Scarce set	$3L$	$300L$
Small set	$10L$	$1000L$
Medium set	$50L$	$5000L$

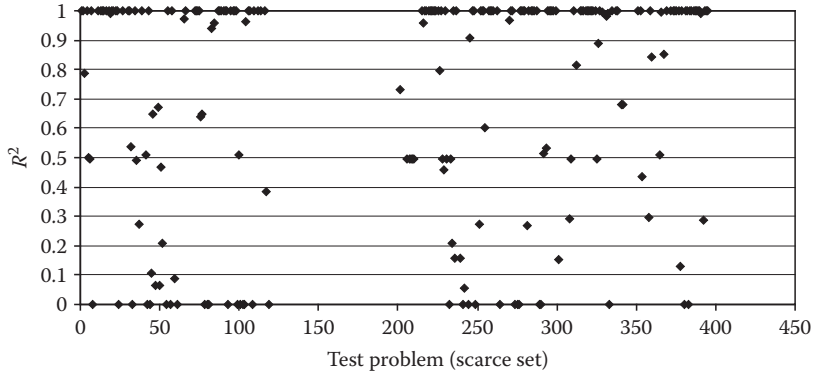

FIGURE 10.22
R^2 metric for the scarce set of training points.

Figure 10.22 shows the R^2 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 excellent interpolation, with $R^2 = 1$.

Figure 10.23 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 less than 1 s, using an AMD Opteron 1.6 GHz processor and 1GB registered ECC DDR PC-2700 RAM. In fact,

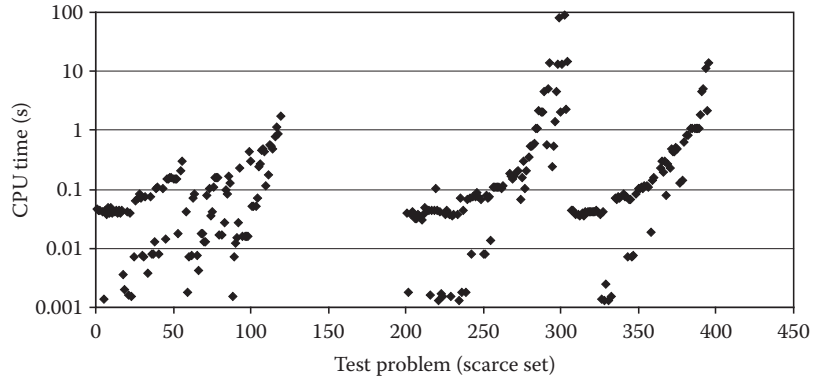

FIGURE 10.23
CPU time for the scarce set of training points.

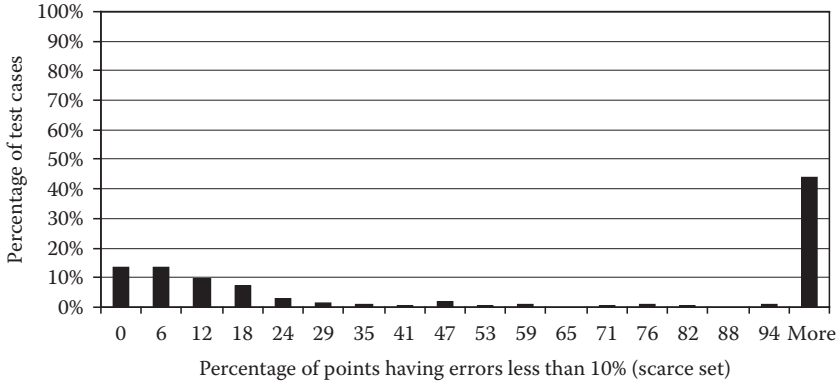

FIGURE 10.24
Testing points with less than 10% error, for the scarce set of training points.

the highest dimensional test cases, which had 100 variables, required only 100 s to be interpolated.

Although the R^2 might indicate some performance behavior of the interpolation function, we decided to use a different measure of accuracy. Figure 10.24 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 10.25 shows the R^2 metric for the small set of training points. Compared to Figure 10.22, it can be seen that the points move toward the value of $R^2 = 1.0$, showing that the accuracy of the interpolation gets better when the number of training points increase.

Figure 10.26 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 s, in almost all test cases, the CPU time is low.

Figure 10.27 shows the percentage of points having errors lower than 10%. Comparing with Figure 10.24, one can see that increasing the number of training points from 3 L

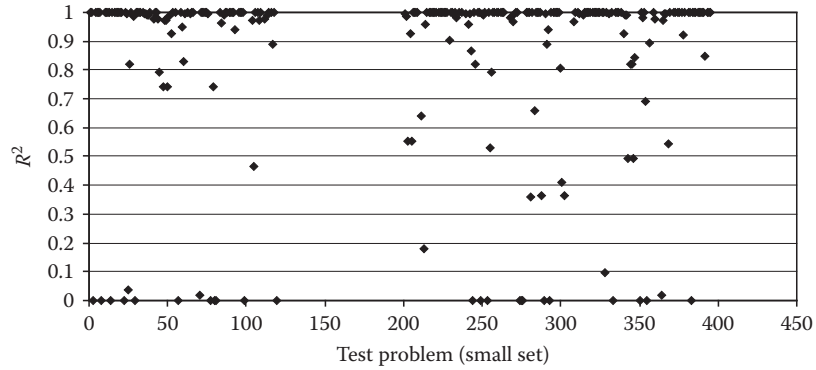

FIGURE 10.25
R^2 metric for the small set of training points.

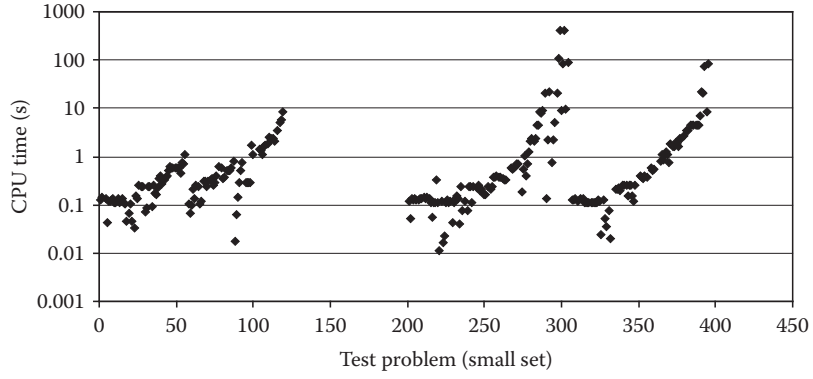

FIGURE 10.26
CPU time for the small set of training points.

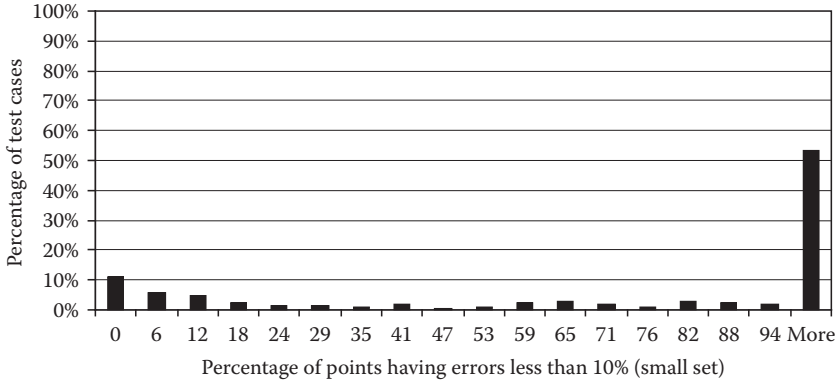

FIGURE 10.27
Testing points with less than 10% error, for the small set of training points.

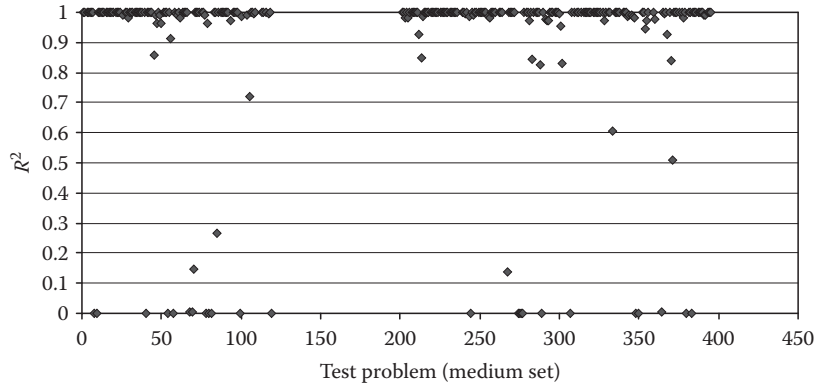

FIGURE 10.28
R^2 metric for the medium set of training points.

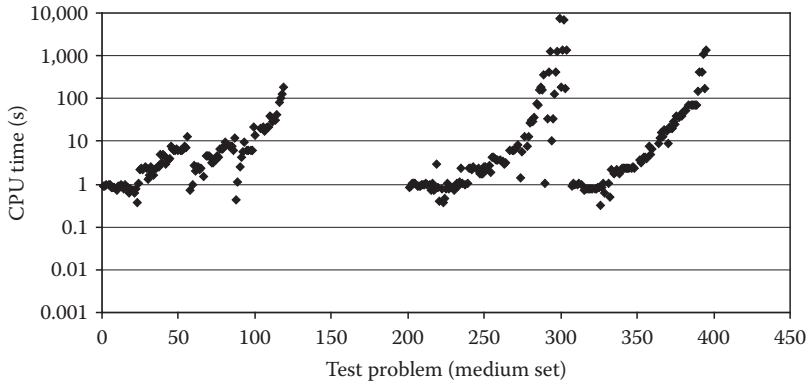

FIGURE 10.29
CPU time for the medium set of training points.

(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 10.28 through 10.30 show the results when a medium set of training points are used.

From Figure 10.28, one can notice that the majority of the test cases have the R^2 metric close to 1.0, indicating a very good interpolation, for a not so large CPU time, as it can be verified at Figure 10.29. From Figure 10.30, 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 metamodel in an optimization task, where the objective function takes too long to be calculated. Thus, instead of optimizing the original function, an interpolation can be used, significantly reducing the computational time.

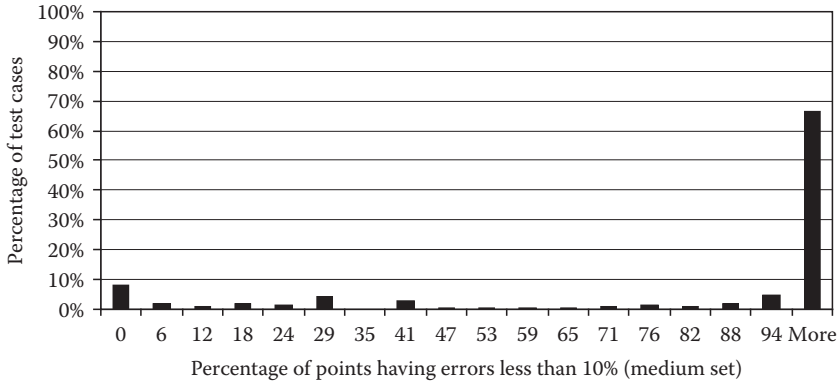

FIGURE 10.30
Testing points with less than 10% error, for the medium set of training points.

10.7 Hybrid Methods with Response Surfaces and Examples

Once the response surface methodology and the hybrid optimizer idea were presented, we will combine both of the sections. This method, called hybrid optimizer ***H2*** (Colaço and Dulikravich, 2007), is quite similar to the ***H1*** presented in Section 10.5, except for the fact that it uses a response surface method at some point of the optimization task. The global procedure is illustrated in Figure 10.31. 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

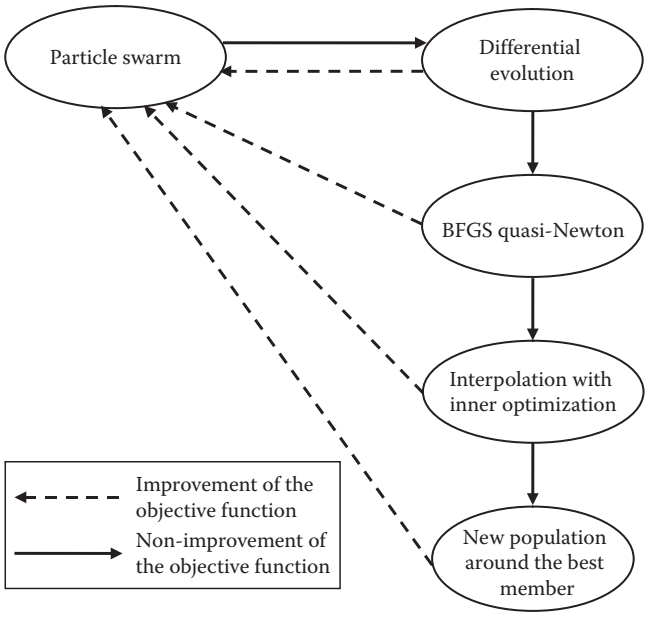

FIGURE 10.31
Global procedure for the hybrid optimization method ***H2***.

function as closely as possible. New values of the objective function are then obtained very cheaply by interpolating their values from the response surface.

In Figure 10.31, if the BFGS cannot find any better solution, the algorithm uses an RBF 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 in the following:

1. Generate an initial population, using the real function (not the interpolated one) $f(\mathbf{P})$. Call this population $\mathbf{\Pi}_{\text{real}}$.
2. Determine the individual that has the minimum value of the objective function, over the entire population $\mathbf{\Pi}_{\text{real}}$ and call this individual $\mathbf{P}_{\text{best}}$.
3. Determine the individual that is more distant from the $\mathbf{P}_{\text{best}}$, over the entire population $\mathbf{\Pi}_{\text{real}}$. Call this individual $\mathbf{P}_{\text{far}}$.
4. Generate a response surface, with the methodology at Section 10.6, using the entire population $\mathbf{\Pi}_{\text{real}}$ 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 10.5, and call the optimum variable of the interpolated function as $\mathbf{P}_{\text{int}}$. 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 $\mathbf{\Pi}_{\text{real}}$ in order to not extrapolate the response surface.
6. If the real objective function $f(\mathbf{P}_{\text{int}})$ is better than all objective functions of the population $\mathbf{\Pi}_{\text{real}}$, replace $\mathbf{P}_{\text{far}}$ by $\mathbf{P}_{\text{int}}$. Otherwise, generate a new individual, using the Sobol's pseudorandom number sequence generator (Sobol and Levitan, 1976) within the upper and lower bounds of the variables, and replace $\mathbf{P}_{\text{far}}$ by this new individual.
7. If the optimum is achieved, stop the procedure. Otherwise, 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 find 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 (Sandgren, 1977), which has 625 local minima and 4 variables. Such function is defined as

$$S(\mathbf{P}) = \sin^2(\pi - z_1) + \sum_{i=1}^{n-1} (z_i - 1)^2 \left[1 = 10\,\sin^2(\pi\, z_{i+1})\right] + (z_4 - 1)^2 \tag{10.51}$$

where

$$z_i = 1 + \frac{P_i - 1}{4} \quad (i = 1, 4) \tag{10.52}$$

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 10.32 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.

From Figure 10.32, 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

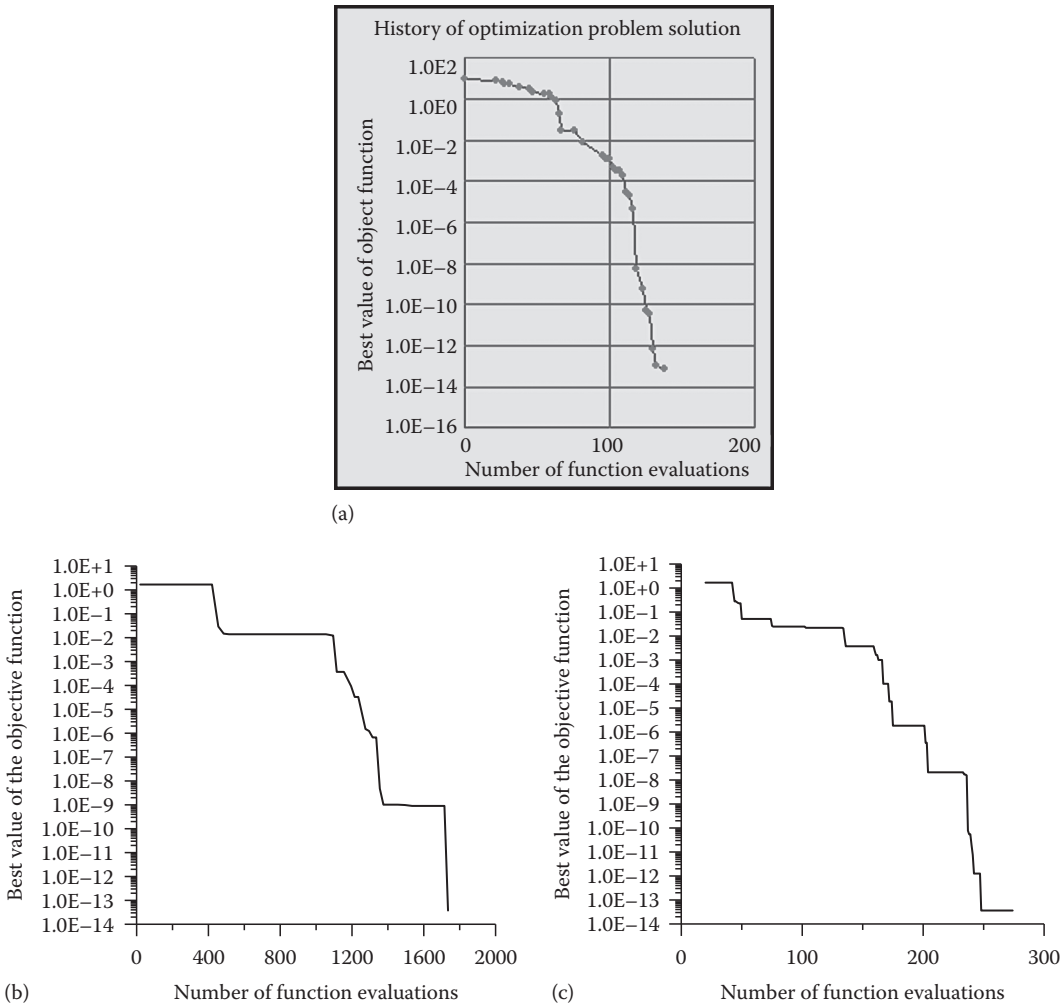

FIGURE 10.32
Optimization history of the Levy #9 function for the (a) IOSO, (b) ***H1***-best, (c) ***H2***-best,

(*continued*)

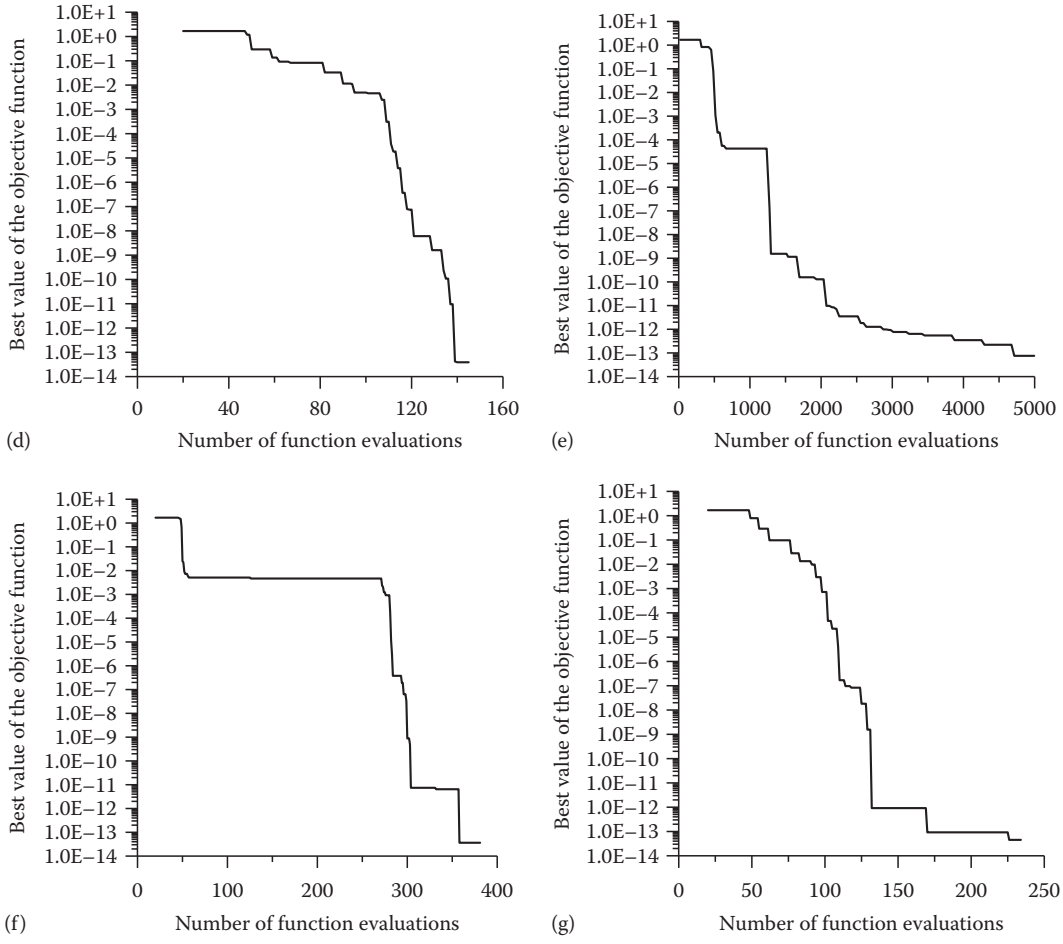

FIGURE 10.32 (continued)
(d) ***H3***-best, (e) ***H1***-worst, (f) ***H2***-worst, and (g) ***H3***-worst optimizers.

reasonably good performance. It is interesting to note that the ***H1*** code is the only one that does not have a response surface model implemented.

The second function tested was the Griewank function (Sandgren, 1977), which is defined as

$$S(\mathbf{P}) = \sum_{i=1}^{n} \frac{P_i^2}{4000} - \prod_{i-1}^{n} \cos\left(\frac{P_i}{\sqrt{i}}\right) + 1; \quad P_i \in \,]-600, 600[\quad (i = 1, 2) \tag{10.53}$$

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

Figure 10.33 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.

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 pointing out that, 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 (More et al., 1981), which is defined as

$$S(P_1, P_2) = 100\left(P_2 - P_1^2\right)^2 + (1 - P_1)^2 \tag{10.54}$$

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 10.34 shows the optimization history of the IOSO, ***H1***, ***H2***, and ***H3*** 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

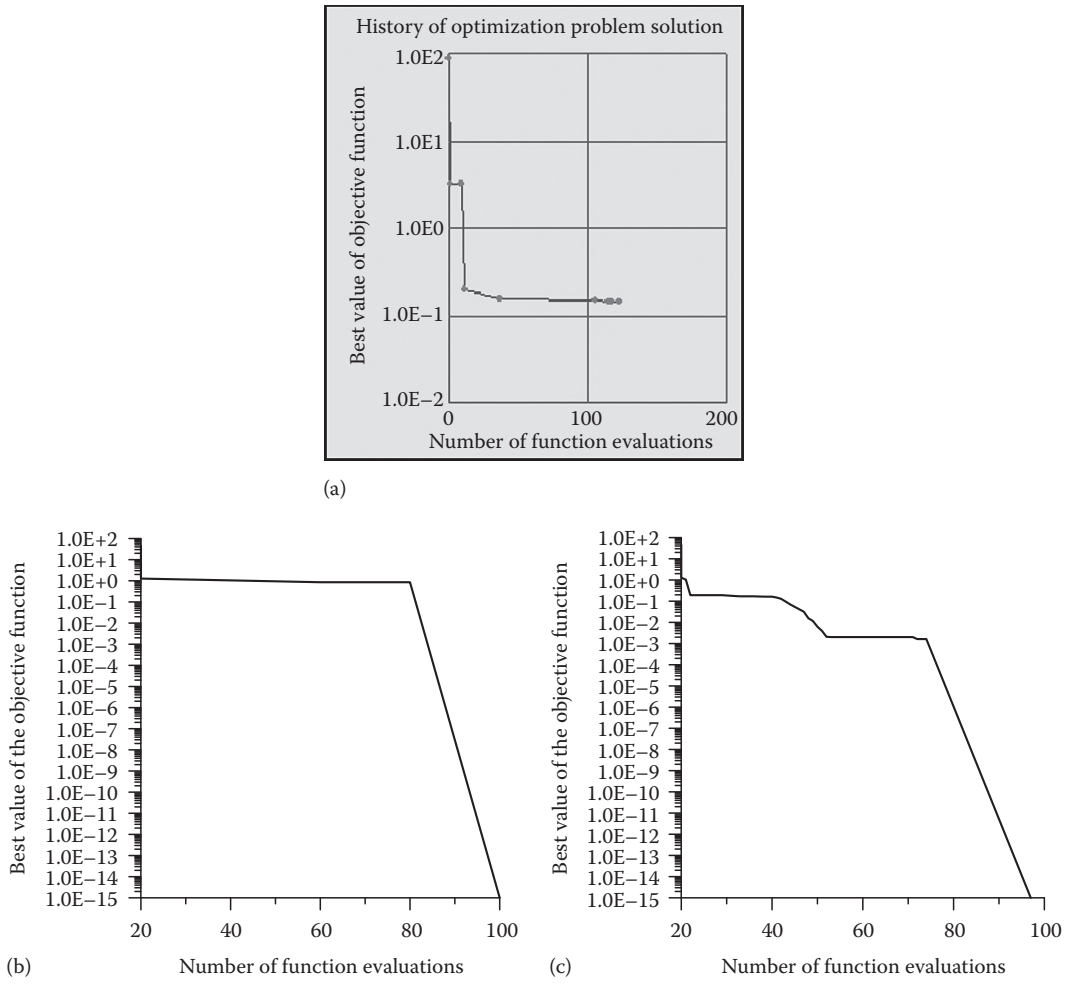

FIGURE 10.33
Optimization history of the Griewank function for the (a) IOSO, (b) ***H1***-best, (c) ***H2***-best,

(*continued*)

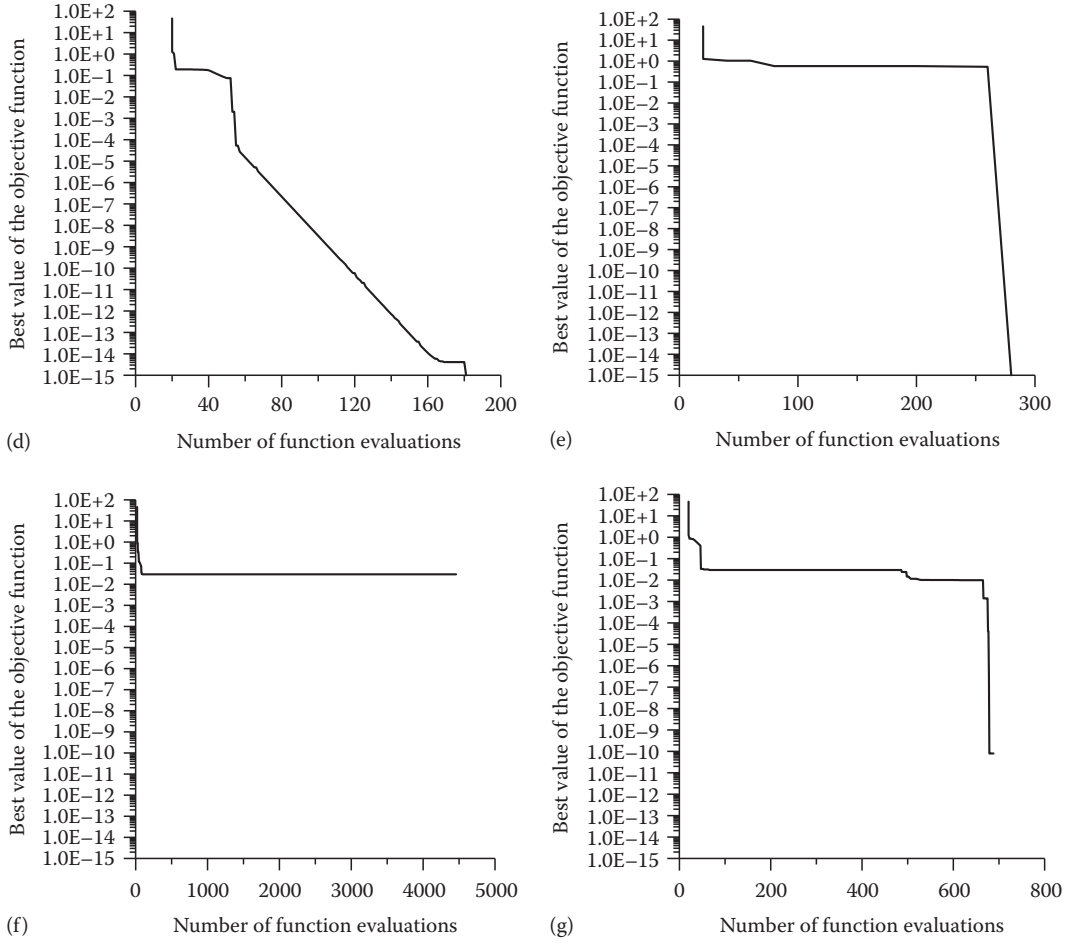

FIGURE 10.33 (continued)
(d) ***H3***-best, (e) ***H1***-worst, (f) ***H2***-worst, and (g) ***H3***-worst optimizers.

inadequately 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 differed 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 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 (Miele and Cantrell, 1969), which is defined as

$$S(\mathbf{P}) = \left[\exp^{(P_1 - P_2)}\right]^4 + 100(P_2 - P_3)^6 + \arctan^4(P_3 - P_4) + P_1^2 \tag{10.55}$$

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 10.35 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.

For this function, the IOSO code was the best, followed by the ***H3***. The ***H2*** code performed very inadequately again. The ***H1*** was able to get to the global minimum after a huge number of objective function calculations. As occurred with the Rosenbrook function, in spite of the fact that ***H3*** results for the objective function differ from the IOSO code, the final values of the variables were $P_1 = 4.0981 \times 10^{-8}$, $P_2 = 0.9864$,

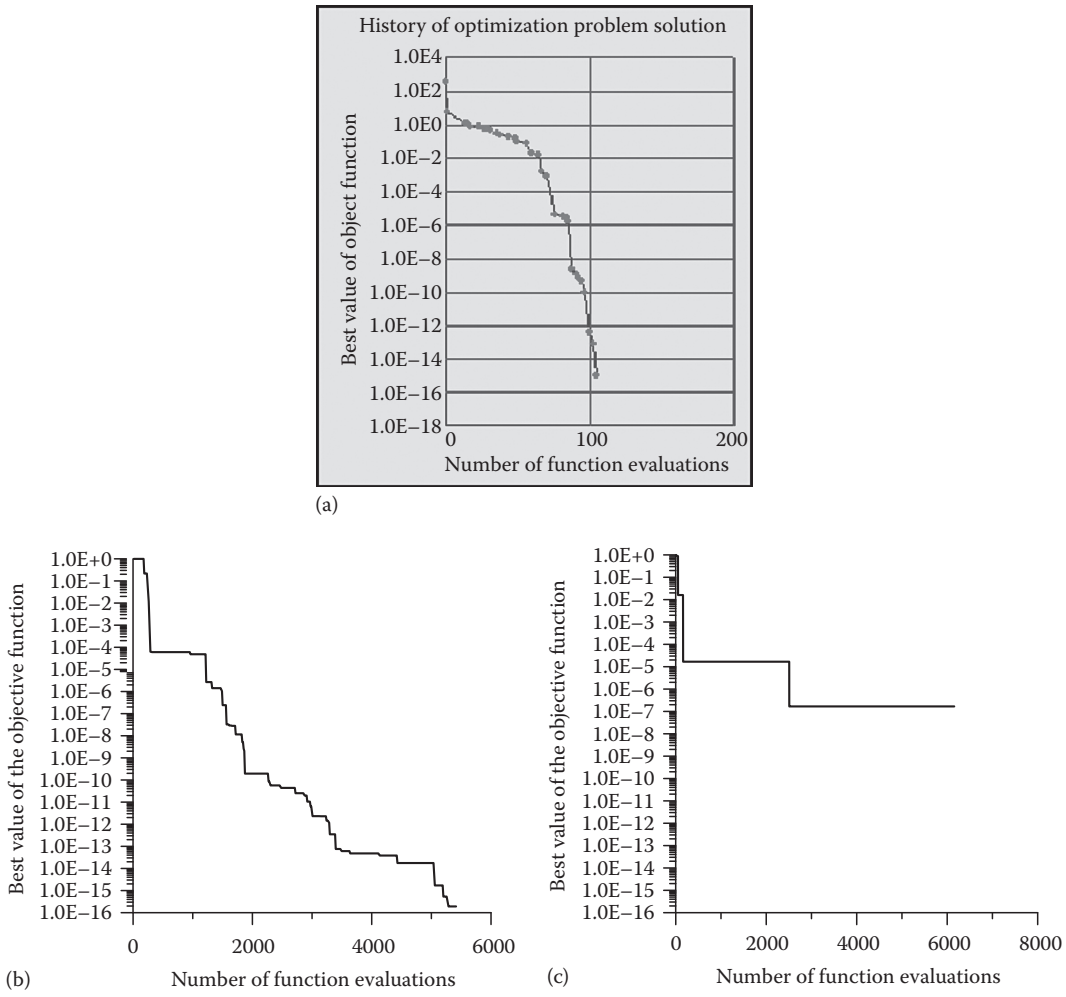

FIGURE 10.34
Optimization history of the Rosenbrook function for the (a) IOSO, (b) ***H1***-best, (c) ***H2***-best,

(*continued*)

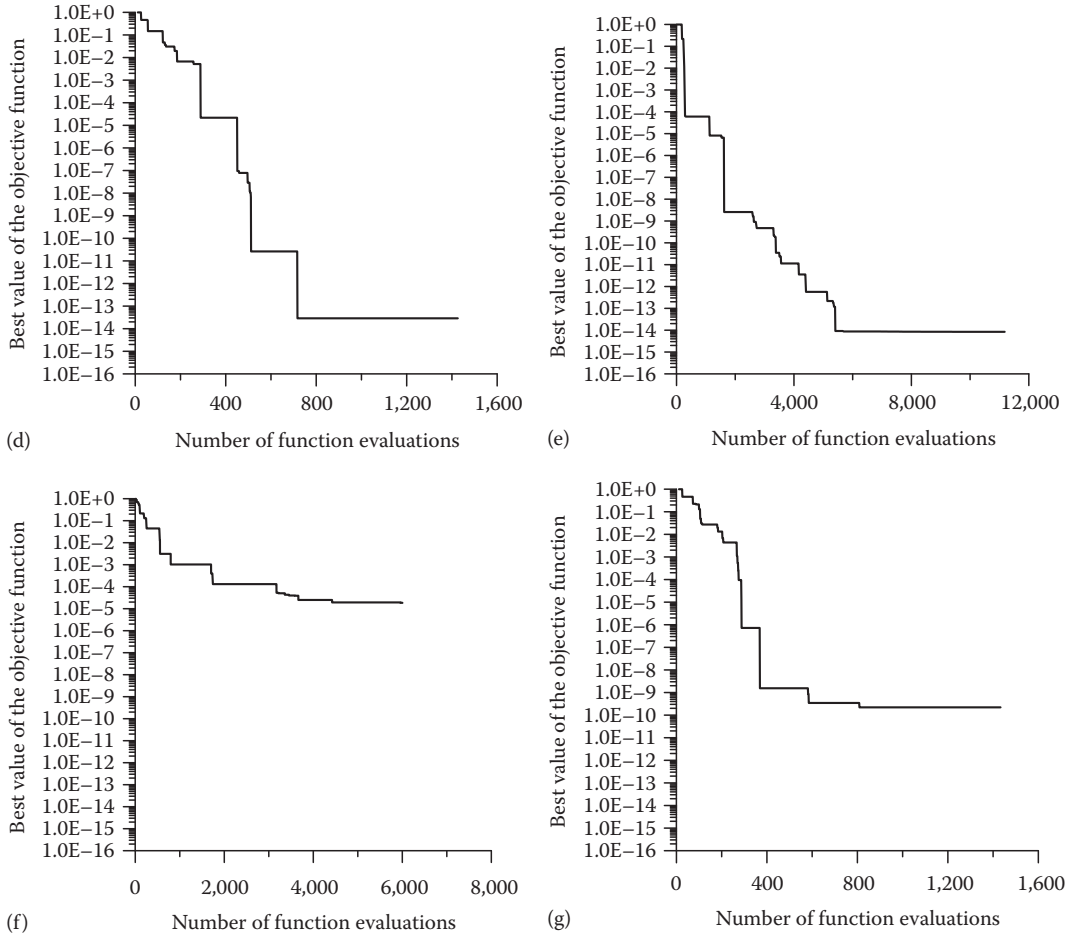

FIGURE 10.34 (continued)
(d) ***H3***-best, (e) ***H1***-worst, (f) ***H2***-worst, and (g) ***H3***-worst optimizers.

$P_3 = 0.9688$, and $P_4 = 0.9626$ for the ***H3*** optimizer and $P_1 = -0.1216 \times 10^{-5}$, $P_2 = 1.002$, $P_3 = 0.9957$, and $P_4 = 0.9962$ for the IOSO code.

10.8 Conclusion

In this chapter, 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 powerful response surface methodology. The combination of the techniques presented here can be used in very complex engineering problems, which demand thousands of objective function calculations.

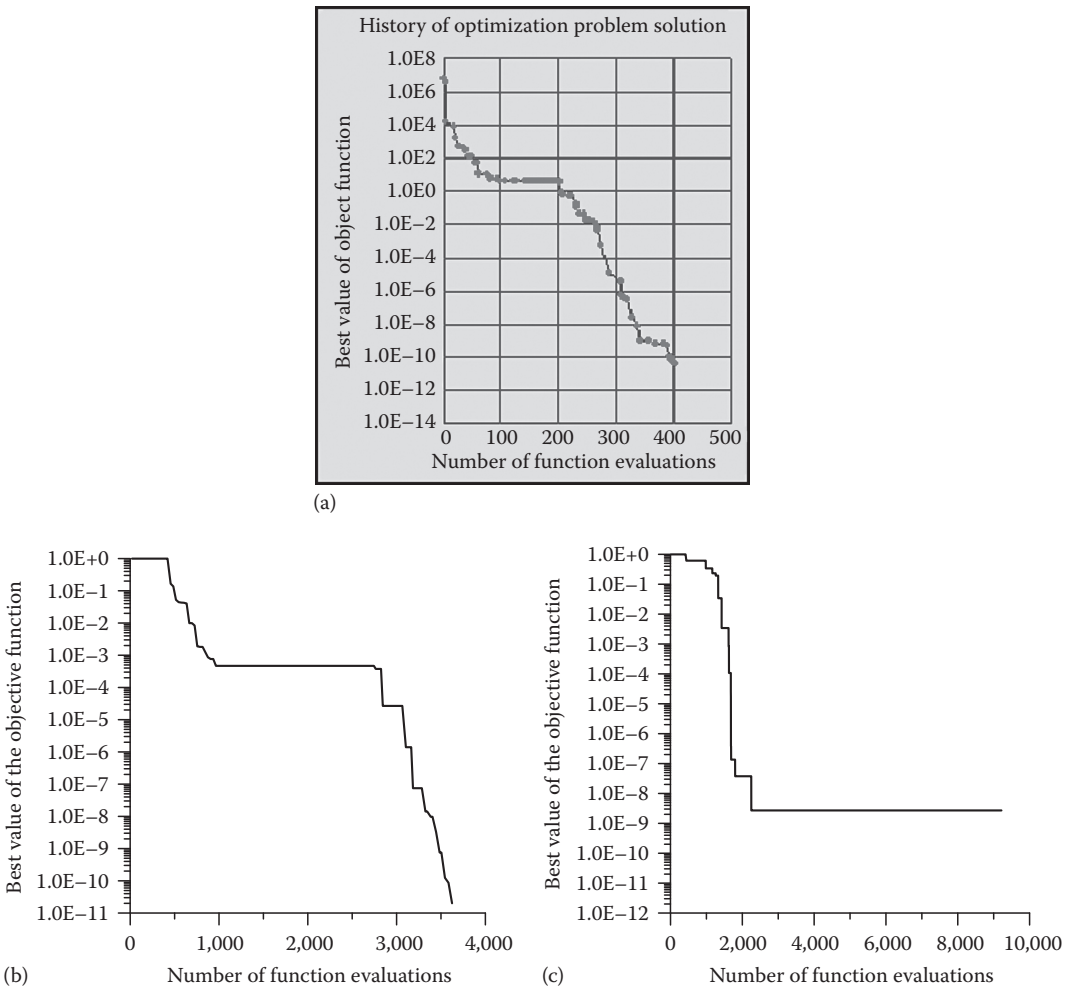

FIGURE 10.35
Optimization history of the Griewank function for the (a) IOSO, (b) ***H1***-best, (c) ***H2***-best,

(*continued*)

Acknowledgments

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 Professor Alain J. Kassab from the 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 grateful for the hospitality of George and Ellen Dulikravich during his stay in Miami for several periods of time during the years 2006, 2007, 2008, and 2009.

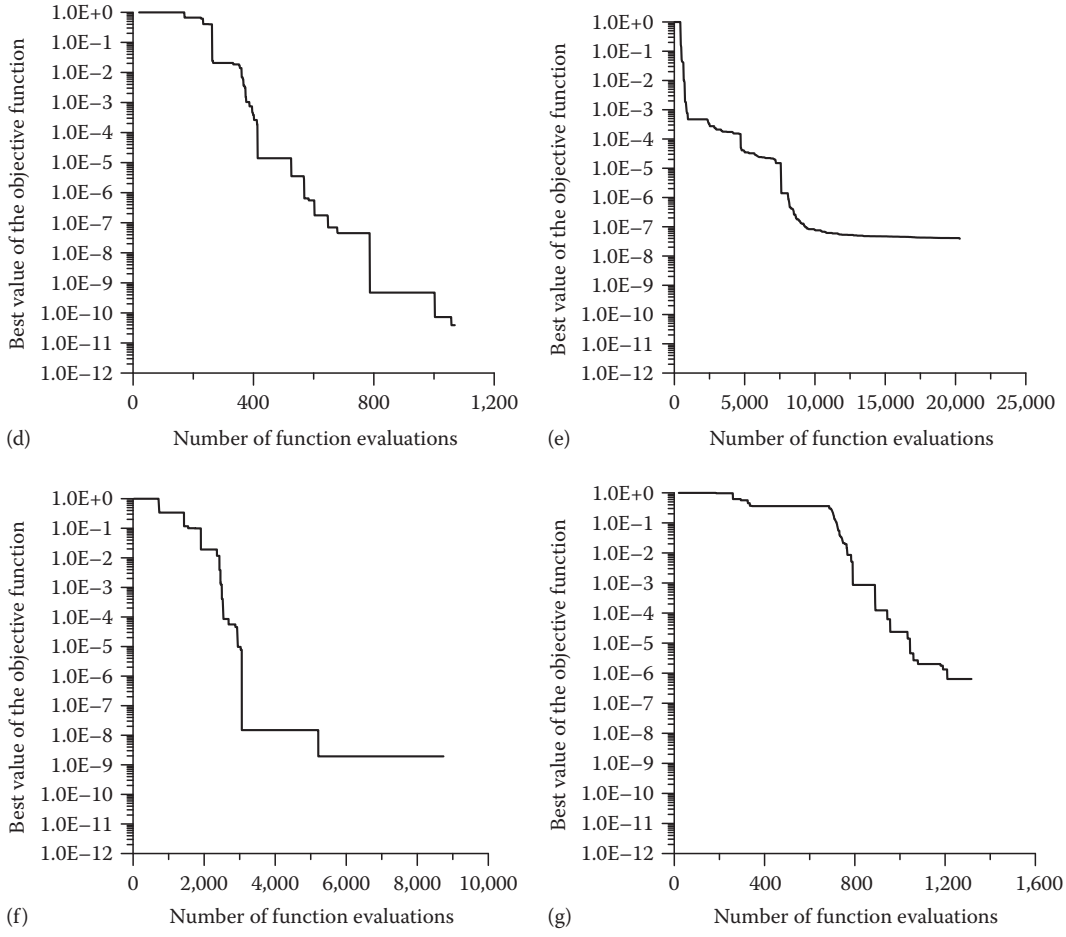

FIGURE 10.35 (continued)
(d) ***H3***-best, (e) ***H1***-worst, (f) ***H2***-worst, and (g) ***H3***-worst optimizers.

Nomenclature

A	approximation of the Hessian
CR	crossover constant
d	direction of descent
E	energy state
F	weight constant which defines the mutation
G	equality constraint
H	approximation for the inverse of the Hessian
I	uncertainty interval
I	identity matrix
J	matrix composed by the diagonal elements of **A**
k	counter for the number of iterations
K	Boltzmann constant

$\mathbf{M}, \mathbf{N}$	auxiliary matrices for the quasi-Newton methods
N	number of parameters (variables)
$\mathbf{P}$	vector of parameters (variables) of the objective function S
q	iteration number for the restraint strategy in the Conjugate Gradient Method
Q	inequality constraint
$\mathbf{r}_1, \mathbf{r}_2$	random number vectors
S	objective function
T	temperature
$\mathbf{Y}$	auxiliary vector for the quasi-Newton methods

Greeks

α	search step size
$\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$	vectors of parameters used in the differential evolution method
δ	delta Dirac function
γ, ψ	conjugation coefficients
λ	auxiliary parameter for the Levenberg-Marquardt method
$\boldsymbol{\pi}_\mathbf{I}$	best value of some individual
$\boldsymbol{\pi}_\mathbf{g}$	best value of the population
$\boldsymbol{\Pi}$	population matrix

References

Alencar Jr., J.P., Orlande, H.R.B., and Ozisik, M.N. 1998. A generalized coordinates approach for the solution of inverse heat conduction problems, in: *11th International Heat Transfer Conference*, Korea, Vol. 7, pp. 53–58, August.

Alifanov, O.M. 1974. Solution of an inverse problem of heat conduction by iteration methods. *Journal of Engineering Physics*, 26(4):471–475.

Alifanov, O.M. 1994. *Inverse Heat Transfer Problems*, Springer-Verlag, New York.

Alifanov, O.M., Artyukhin, E., and Rumyantsev, A. 1995. *Extreme Methods for Solving Ill-Posed Problems with Applications to Inverse Heat Transfer Problems*, Begell House, New York.

Artyukhin, E.A. 1993. Iterative algorithms for estimating temperature-dependent thermophysical characteristics, in: *First International Conference on Inverse Problems in Engineering: Theory and Practice*, eds. N. Zabaras, K. Woodburry, and M. Raynaud, pp. 101–108, Palm Coast, FL, June.

Bard, Y.B. 1974. *Nonlinear Parameter Estimation*, Academic Press, New York.

Beale, E.M.L. 1972. A derivation of conjugate gradients, in: *Numerical Methods for Nonlinear Optimization*, ed. F.A. Lootsma, pp. 39–43, Academic Press, New York.

Beck, J.V. 1999. Sequential methods in parameter estimation, in: *Third International Conference on Inverse Problems in Engineering*, Tutorial Session, Port Ludlow, WA. AQ10

Beck, J.V. and Arnold, K.J. 1977. *Parameter Estimation in Engineering and Science*, Wiley Interscience, New York.

Beck, J.V., Blackwell, B., and St. Clair, C.R. 1985. *Inverse Heat Conduction: Ill-Posed Problems*, Wiley Interscience, New York.

Belegundu, A.D. and Chandrupatla, T.R. 1999. *Optimization Concepts and Applications in Engineering*, Prentice Hall, Denver, CO.

Broyden, C.G. 1965. A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 19:577–593.

Broyden, C.G. 1967. Quasi-Newton methods and their applications to function minimization. *Mathematics of Computation*, 21:368–380.

Buhmann, M.D. 2003. Radial basis functions on grids and beyond, in: *International Workshop on Meshfree Methods*, Lisbon, Portugal.

Colaço, M.J. and Dulikravich, G.S. 2006. A multilevel hybrid optimization of magnetohydrodynamic problems in double-diffusive fluid flow. *Journal of Physics and Chemistry of Solids*, 67:1965–1972.

Colaço, M.J. and Dulikravich, G.S. 2007. Solidifcation of double-diffusive flows using thermo-magneto-hydrodynamics and optimization. *Materials and Manufacturing Processes*, 22:594–606.

Colaço, M.J., Dulikravich, G.S., and Martin, T.J. 2004. Optimization of wall electrodes for electro-hydrodynamic control of natural convection during solidification. *Materials and Manufacturing Processes*, 19(4):719–736.

Colaço, M.J., Dulikravich, G.S., and Martin, T.J. 2005a. Control of unsteady solidification via optimized magnetic fields. *Materials and Manufacturing Processes*, 20(3):435–458.

Colaço, M.J., Dulikravich, G.S., Orlande, H.R.B., and Martin, T.J. 2005b. Hybrid optimization with automatic switching among optimization algorithms, in: *Evolutionary Algorithms and Intelligent Tools in Engineering Optimization*, eds. W. Annicchiarico, J. Periaux, M. Cerrolaza, and G. Winter, pp. 92–118, WIT Press/Computational Mechanics, Southampton, U.K.

Colaço, M.J., Dulikravich, G.S., and Sahoo, D. 2007. A comparison of two methods for fitting high dimensional response surfaces, in: *Proceedings of International Symposium on Inverse Problems, Design and Optimization (IPDO-2007)*, eds. G.S. Dulikravich, H.R.B. Orlande, M. Tanaka, and M. J. Colaço, Miami Beach, FL, April 16–18.

Colaço, M.J., Dulikravich, G.S., and Sahoo, D. 2008. A response surface method based hybrid optimizer. *Inverse Problems in Science and Engineering*, 16:717–742.

Colaço, M.J. and Orlande, H.R.B. 1998. Estimation of the heat transfer coefficient at the surface of a plate by using the conjugate gradient method, in: *VII Brazilian National Meeting of Thermal Sciences— ENCIT*, Rio de Janeiro, Brazil, Vol. 1, pp. 189–194.

Colaço, M.J. and Orlande, H.R.B. 1999. A comparison of different versions of the conjugate gradient method of function estimation. *Numerical Heat Transfer, Part A*, 36(2):229–249.

Colaço, M.J. and Orlande, H.R.B. 2000. 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*, eds. M. Tanaka and G. S. Dulikravich, pp. 409–418, Nagano, Japan.

Colaço, M.J. and Orlande, H.R.B. 2001a. 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.

Colaço, M.J. and Orlande, H.R.B. 2001b. Inverse forced convection problem of simultaneous estimation of two boundary heat fluxes in irregularly shaped channels. *Numerical Heat Transfer, Part A*, 39:737–760.

Colaço, M.J. and Orlande, H.R.B. 2002. A natural convection inverse problem of simultaneous identification of two boundary heat fluxes in rectangular cavities, in: *12th International Heat Transfer Conference*, Grenoble, France.

Colaço, M.J. and Orlande, H.R.B. 2004. Inverse natural convection problem of simultaneous estimation of two boundary heat fluxes in irregular cavities. *International Journal of Heat and Mass Transfer*, 47:1201–1215.

Colaço, M.J., Orlande, H.R.B., and Dulikravich, G.S. 2006. Inverse and optimization problems in heat transfer. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 28(1):1–24.

Corana, A., Marchesi, M., Martini, C., and Ridella, S. 1987. Minimizing multimodal functions of continuous variables with the 'simulated annealing algorithm.' *ACM Transactions on Mathematical Software*, 13:262–280.

Daniel, J.W. 1971. *The Approximate Minimization of Functionals*, Prentice-Hall, Englewood Cliffs, NJ.

Dantas, L.B. and Orlande, H.R.B. 1996. A function estimation approach for determining temperature-dependent thermophysical properties. *Inverse Problems in Engineering*, 3:261–279.

Darwin, C. 1859. *On the Origin of Species*, John Murray, London, U.K.

Davidon, W.C. 1959. *Variable Metric Method for Minimization*, Argonne National Laboratory, ANL-5990, Argonne, IL.

Deb, K. 2002. *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, New York.

de Boor, C. 1978. *A Practical Guide to Splines*, Springer-Verlag, New York.
Denisov, A.M. 1999. *Elements of the Theory of Inverse Problems*, VSP, the Netherlands.
Dennis, J. and Schnabel, R. 1983. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall, New York.
Dulikravich, G.S. and Colaço, M.J. 2006. Convective heat transfer control using magnetic and electric fields. *Journal of Enhanced Heat Transfer*, 13(2):139–155.
Dulikravich, G.S., Colaço, M.J., Dennis, B.H., Marting, T.J., Egorov, I.N., and Lee, S. 2004. Optimization of intensities, and orientations of magnets controlling melt flow during solidification. *Materials and Manufacturing Processes*, 19(4):695–718.
Dulikravich, G.S., Colaço, M.J., Martin, T.J., and Lee, S. 2003. An inverse method allowing user-specified layout of magnetized micro-fibers in solidifying composites. *Journal of Composite Materials, UK*, 37(15):1351–1365.
Dulikravich, G.S., Egorov, I.N., and Colaço, M.J. 2008. Optimizing chemistry of bulk metallic glasses for improved thermal stability. *Modelling and Simulation in Materials Science and Engineering*, 16:075010.
Dulikravich, G.S. and Martin, T.J. 1994. Inverse design of super-elliptic cooling passages in coated turbine blade airfoils. *AIAA Journal of Thermophysics and Heat Transfer*, 8(2):288–294.
Dulikravich, G.S. and Martin, T.J. 1996. Inverse shape and boundary condition problems and optimization in heat conduction, in: *Advances in Numerical Heat Transfer*, eds. W.J. Minkowycz and E. M. Sparrow, Chapter 10, pp. 381–426, Taylor & Francis, Boca Raton, FL.
Dulikravich, G.S., Martin, T.J., Dennis, B.H., and Foster, N.F. 1999. Multidisciplinary hybrid constrained GA optimization, Invited lecture, 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, Chapter 12, pp. 231–260, John Wiley & Sons, Jyvaskyla, Finland, May 30–June 3.
Eberhart, R., Shi, Y., and Kennedy, J. 2001. *Swarm Intelligence*, Morgan Kaufmann, San Francisco, CA.
Fletcher, R. 2000. *Practical Methods of Optimization*, John Wiley & Sons, New York.
Fletcher, R. and Powell, M.J.D. 1963. A rapidly convergent descent method for minimization. *Computer Journal*, 6:163–168.
Fletcher, R. and Reeves, C.M. 1964. Function minimization by conjugate gradients. *Computer Journal*, 7:149–154.
Fox, R.L. 1971. *Optimization Methods for Engineering Design*, Addison-Wesley Publishing Company, Reading, MA. AQ11
Goffe, W.L., Ferrier, G.D., and Rogers, J. 1994. Global optimization of statistical functions with simulated annealing. *Journal of Econometrics*, 60:65–99.
Goldberg, D.E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, Reading, MA.
Hadamard, J. 1923. *Lectures on Cauchy's Problem in Linear Differential Equations*, Yale University Press, New Haven, CT.
Hardy, R.L. 1971. Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysics Research*, 176:1905–1915.
Hensel, E. 1991. *Inverse Theory and Applications for Engineers*, Prentice Hall, Englewood Cliffs, NJ.
Hestenes, M.R. and Stiefel, E. 1952. Method of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards. Section B*, 49:409–436.
Hock, W. and Schittkowski, K. 1981. *Test Examples for Nonlinear Programming Codes, Lecture Notes in Economics and Mathematical Systems, 187*, Springer-Verlag, Berlin/Heidelberg/New York.
Huang, C.H. and Tsai, C.H. 1997. A shape identification problem in estimating time-dependent irregular boundary configurations, in: *National Heat Transfer Conference*, ASME HTD-340, Vol. 2, eds. G.S. Dulikravich and K.A. Woodbury, pp. 41–48.
IOSO NM Version 1.0, User's Guide. 2003. IOSO Technology Center, Moscow, Russia.
Isakov, V. 1998. *Inverse Problems for Partial Differential Equations, Applied Mathematical Sciences*, Vol. 127, Springer, New York.
Jaluria, Y. 1998. *Design and Optimization of Thermal Systems*, McGraw Hill, New York.

Jarny, Y., Ozisik, M.N., and Bardon, J.P. 1991. A general optimization method using adjoint equation for solving multidimensional inverse heat conduction. *International Journal of Heat and Mass Transfer*, 34(11):2911–2919.

Jin, R., Chen, W., and Simpson, T.W. 2000. Comparative studies of metamodeling techniques under multiple modeling criteria, in: *Proceedings of the 8th AIAA/USAF/NASA/ISSMO Multidisciplinary Analysis & Optimization Symposium*, AIAA 2000-4801, Long Beach, CA, September 6–8.

Kansa, E.J. 1990. Multiquadrics—A scattered data approximation scheme with applications to computational fluid dynamics—II: Solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers and Mathematics with Applications*, 19:149–161.

Kaufman, M., Balabanov, V., Burgee, S.L., Giunta, A.A., Grossman, B., Mason, W.H., and Watson, L.T. 1996. Variable complexity response surface approximations for wing structural weight in HSCT design, AIAA Paper 96-0089, in: *Proceedings of the 34th Aerospace Sciences Meeting and Exhibit*, Reno, NV.

Kennedy, J. 1999. Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance, in: *Proceedings of the 1999 Congress of Evolutionary Computation*, IEEE Press, Vol. 3, pp. 1931–1938. AQ12

Kennedy, J. and Eberhart, R.C. 1995. Particle swarm optimization, in: *Proceedings of the 1995 IEEE International Conference on Neural Networks*, Perth, Australia, Vol. 4, pp. 1942–1948.

Kirsch, A. 1996. *An Introduction to the Mathematical Theory of Inverse Problems, Applied Mathematical Sciences*, Vol. 120, Springer, New York.

Kurpisz, K. and Nowak, A.J. 1995. *Inverse Thermal Problems*, WIT Press, Southampton, U.K.

Lancaster, P. and Salkauskas, K. 1986. *Curve and Surface Fitting: An Introduction*, Academic Press, Harcourt Brace Jovanovic, New York.

Leitão, V.M.A. 2001. A mesheless method for Kirchhoff plate bending problems. *International Journal of Numerical Methods in Engineering*, 52:1107–1130.

Leitão, V.M.A. 2004. RBF-based meshless methods for 2D elastostatic problems. *Engineering Analysis with Boundary Elements*, 28:1271–1281.

Levenberg, K. 1944. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168.

Machado, H.A. and Orlande, H.R.B. 1997. 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.

Madala, H.R. and Ivakhnenko, A.G. 1994. *Inductive Learning Algorithms for Complex Systems Modeling*, CRC Press, Boca Raton, FL.

Marquardt, D.W. 1963. An algorithm for least squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11:431–441.

Miele, A. and Cantrell, J.W. 1969. Study on a memory gradient method for the minimization of functions. *Journal of Optimization, Theory and Applications*, 3(6):459–470.

Moral, R.J. and Dulikravich, G.S. 2008. A hybrid self-organizing response surface methodology, Paper AIAA-2008–5891, in: *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, BC, Canada, September 10–12.

Moré, J.J. 1977. The Levenberg-Marquardt algorithm: Implementation and theory, in: *Numerical Analysis, Lecture Notes in Mathematics*, ed. G.A. Watson, Vol. 630, pp. 105–116, Springer-Verlag, Berlin.

More, J.J., Gabow, B.S., and Hillstrom, K.E. 1981. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, 17–41. AQ13

Morozov, V.A. 1984. *Methods for Solving Incorrectly Posed Problems*, Springer Verlag, New York.

Murio, D.A. 1993. *The Mollification Method and the Numerical Solution of Ill-Posed Problems*, Wiley Interscience, New York.

Naka, S., Yura, T.G., and Fukuyama, T. 2001. Practical distribution state estimation using hybrid particle swarm optimization, in: *Proceedings IEEE Power Engineering Society*, Winter Meeting, Columbus, OH, January 28–February 1.

Oliver, M.A. and Webster, R. 1990. Kriging: A method of interpolation for geographical information system. *International Journal of Geographical Information Systems*, 4(3):313–332.

Orlande, H.R.B., Colaço, M.J., and Malta, A.A. 1997. Estimation of the heat transfer coefficient in the spray cooling of continuously cast slabs, in: *National Heat Transfer Conference*, ASME HTD-340, Vol. 2, eds. G.S. Dulikravich and K.A. Woodbury, pp. 109–116, June.

Ozisik, M.N. and Orlande, H.R.B. 2000. *Inverse Heat Transfer: Fundamentals and Applications*, Taylor & Francis, New York.

Padilha, R.S., Santos, H.F.S., Colaço, M.J., and Cruz, M.E.C. 2009. Single and multi-objective optimization of a cogeneration system using hybrid algorithms. *Heat Transfer Engineering*, 30:261–271.

Polak, E. 1971. *Computational Methods in Optimization*, Academic Press, New York.

Powell, M.J.D. 1977. Restart procedures for the conjugate gradient method. *Mathematical Programming*, 12:241–254.

Ramm, A.G., Shivakumar, P.N., and Strauss, A.V. (eds.). 2000. *Operator Theory and Applications*, American Mathematical Society, Providence, RI.

Sabatier, P.C. (ed.). 1978. *Applied Inverse Problems*, Springer Verlag, Hamburg.

Sahoo, D. and Dulikravich, G.S. 2006. Evolutionary wavelet neural network for large scale function estimation in optimization, in: *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, AIAA Paper AIAA-2006–6955, Portsmouth, VA, September 6–8.

Sandgren, E. 1977. The utility of nonlinear programming algorithms. PhD thesis, Purdue University, IN.

Schittkowski, K. 1987. *More Test Examples for Nonlinear Programming, Lecture Notes in Economics and Mathematical Systems*, 282, Springer Verlag, New York.

Silva, P.M.P., Orlande, H.R.B., Colaço, M.J., Shiakolas, P.S., and Dulikravich, G.S. 2007. Identification and design of a source term in a two-region heat conduction problem. *Inverse Problems in Science and Engineering*, 15:661–677.

Sobol, I. and Levitan, Y.L. 1976. The production of points uniformly distributed in a multidimensional cube, Preprint *IPM Akad. Nauk SSSR*, No. 40, Moscow, Russia.

Stoecker, W.F. 1989. *Design of Thermal Systems*, McGraw Hill, New York.

Storn, R. and Price, K.V. 1996. Minimizing the real function of the ICEC'96 contest by differential evolution, in: *IEEE Conference on Evolutionary Computation*, pp. 842–844.

Tikhonov, A.N. and Arsenin, V.Y. 1977. *Solution of Ill-Posed Problems*, Winston & Sons, Washington, DC.

Truffart, B., Jarny, J., and Delaunay, D. 1993. A general optimization algorithm to solve 2-d boundary inverse heat conduction problems using finite elements, in: *First International Conference on Inverse Problems in Engineering: Theory and Practice*, eds. N. Zabaras, K. Woodburry, and M. Raynaud, pp. 53–60, Palm Coast, FL, June.

Trujillo, D.M. and Busby, H.R. 1997. *Practical Inverse Analysis in Engineering*, CRC Press, Boca Raton, FL.

Wellele, O., Orlande, H.R.B., Ruberti, N.J., Colaço, M.J., and Delmas, A. 2006. Identification of the thermophysical properties of orthotropic semi-transparent materials, in: *13th International Heat Transfer Conference*, Sydney, Australia.

Wendland, H. 1998. Error estimates for interpolation by compactly supported radial basis functions of minimal degree. *Journal of Approximation Theory*, 93:258–272.

Woodbury, K. 2002. *Inverse Engineering Handbook*, CRC Press, Boca Raton, FL.

Yagola, A.G., Kochikov, I.V., Kuramshina, G.M., and Pentin, Y.A. 1999. *Inverse Problems of Vibrational Spectroscopy*, VSP, the Netherlands.

Zubelli, J.P. 1999. *An Introduction to Inverse Problems: Examples, Methods and Questions*, Institute of Pure and Applied Mathematics, Rio de Janeiro, Brazil.

AUTHOR QUERIES

[AQ1] Please specify whether the reference Colaço et al., 2005 cited here and in subsequent occurrences refer to Colaço et al., 2005a or Colaço et al., 2005b.
[AQ2] Running head is edited from the Chapter title. Please check.
[AQ3] Please provide part figure captions (a) through (d) in Figure 10.1 caption.
[AQ4] Please confirm all the heading levels identified in this chapter for correctness.
[AQ5] Kindly provide the expansion of the acronyms ''RMS, SVD, LU'' if appropriate.
[AQ6] Please confirm the change made in Equation 10.34.
[AQ7] Please add the references Goffet et al., 1944; Hardy, 1991; Belegundu et al., 1999; Eberhart and Kennedy, 2001 to the reference list.
[AQ8] Please confirm the figure number cited in the sentence ''In Figure 10.16, the algorithm . . . ''.
[AQ9] Please check the sentence starting: ''This approximation. . . . .'' for correctness.
[AQ10] Please provide the date of proceedings for Refs. Beck (1999), Buhmann (2003), Colaço and Orlande (1998, 2000, 2002), Kaufman et al., 1996, Kennedy and Eberhart, 1995, Kennedy 1999, Storn and Price 1996, Wellele et al. (2006).
[AQ11] Please provide in-text citation for references Fox, 1971 and Hardy, 1971.
[AQ12] Please provide location of proceedings for Kennedy, 1999.
[AQ13] Please provide volume number for reference More et al. (1981).