EVOLUTIONARY ALGORITHMS AND INTELLIGENT TOOLS
IN ENGINEERING OPTIMIZATION
W. Annicchiarico, J. Périaux, M. Cerrolaza and G. Winter (Eds.)
© CIMNE, Barcelona, Spain 2004

HYBRID OPTIMIZATION WITH AUTOMATIC SWITCHING AMONG OPTIMIZATION ALGORITHMS

Marcelo J. Colaço
Department of Mechanical and Materials Engineering
Military Institute of Engineering (IME)
Pç. Gen. Tiburcio, 80, RJ, 22290-270, BRAZIL
Email: colaco@ime.eb.br
web page: http://www.ime.eb.br

George S. Dulikravich[*]
Department of Mechanical and Materials Engineering
Florida International University (FIU)
10555 West Flagler Street, Miami, FL 33174, USA
Email: dulikrav@fiu.edu
web page: http://www.fiu.edu

Helcio R. B. Orlande
Department of Mechanical Engineering
Federal University of Rio de Janeiro (UFRJ)
Cx. Postal 68503, RJ, 21945-970, BRAZIL
Email: helcio@serv.com.ufrj.br
web page: http://www.ufrj.br

Thomas J. Martin
Turbine Discipline Engineering & Optimization Group
Pratt & Whitney Engine Company
400 Main Street, M/S 169-20,
East Hartford, CT 06108, USA

Abstract. In this chapter we present a hybrid approach to optimization problems, where we use deterministic and stochastic/evolutionary optimization algorithms. The basic idea is to start with a non-determinist method in order to reduce the search-space to a region where the global minima is located. At this point an automatic switch to a deterministic method is performed in order to obtain a rapid convergence to the global extreme. A revision of some well-known optimization algorithms is presented, followed by a comparison among the different optimization techniques and the application of the hybrid method to several mathematical functions having multiple extrema.

Key words: optimization, hybrid methods, search algorithms, evolutionary optimization

1 INTRODUCTION

Classical optimization techniques are typically based on deterministic models as in the case of the various gradient-like methods: Steepest Descent method[1-4], Conjugate Gradient method[1-26], Newton-Raphson method[1-4], quasi Newton methods[1-4] and others. However, in spite of their fast convergence rate, such techniques are strongly dependent on the initial guess used for the optimization task, especially when dealing with problems containing a large number of local minima.

Evolutionary methods try to mimic the behavior of species, such as birds (Particle Swarm method[27-30]), ants (Ant Colony optimization[31]) and others. Several of them, as in the case of the Genetic Algorithm methods[32], try to mimic the Evolutionary Theory of Species, proposed by Darwin[33]. These kinds of methods are more likely to find global minima than the deterministic methods. However, they are, in general, slower than the former ones. The objective of so-called Hybrid Optimization methods[34-38] is to take advantage of the robustness of the evolutionary methods and the fast convergence rate of the deterministic methods. The basic idea is to start with an evolutionary method in order to reduce the large initial search space to a narrow region where the determinist method is then employed to rapidly locate the minima.

2 DETERMINISTIC METHODS

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

2.1. Steepest Descent Method[1-4]

The most basic gradient-based method is the Steepest Descent method. Some of the concepts developed here will be used in the next sections, where we will discuss more advanced methods. The basic idea of this method is to "walk" in the opposite direction of the locally highest variation of the objective function, in order to locate the maximum or minimum value of it. This can be exemplified in Figure 1.

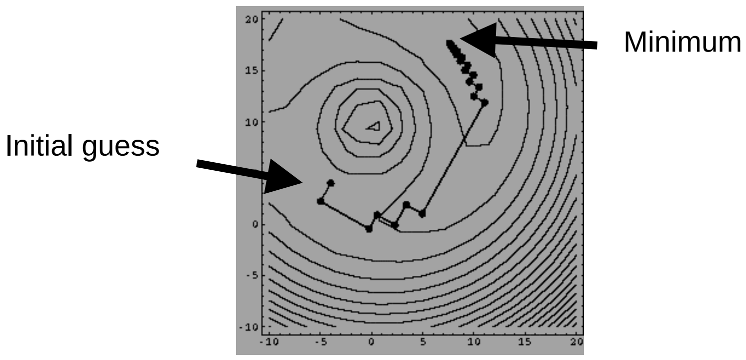


Figure 1: Convergence history for the Steepest Descent method.

The objective function can be mathematically stated as

$$U = U(\mathbf{x}); \quad \mathbf{x} = \{x_1, x_2, ..., x_N\} \tag{1}$$

The direction in which the objective function U varies most rapidly is the direction of gradient of U. Thus, for the case with two variables (Figure 1) the gradient is

$$\nabla U = \frac{\partial U}{\partial x_1}\mathbf{i}_1 + \frac{\partial U}{\partial x_2}\mathbf{i}_2 \tag{2}$$

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

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^{k+1} \tag{3}$$

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

$$\mathbf{d}^{k+1} = -\nabla U(\mathbf{x}^k) \tag{4}$$

In spite of this being the natural choice for the direction of descent, it is not very efficient as can be seen in Figure 1. Usually, the method starts with large variations in the objective function, but as the minimum of the objective function is being reached, 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 (3) and the definition of the objective function (1), we have that at iteration level k + 1

$$U(\mathbf{x}^{k+1}) = U(\mathbf{x}^k + \alpha^k \mathbf{d}^{k+1}) \tag{5}$$

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

$$\frac{dU\left(\mathbf{x}^{k+1}\right)}{d\alpha^k} = 0 \tag{6}$$

Using the chain rule

$$\frac{dU\left(\mathbf{x}^{k+1}\right)}{d\alpha^k} = \frac{dU\left(x_1^{k+1}\right)}{dx_1^{k+1}}\frac{dx_1^{k+1}}{d\alpha^k} + \frac{dU\left(x_2^{k+1}\right)}{dx_2^{k+1}}\frac{dx_2^{k+1}}{d\alpha^k} + \cdots + \frac{dU\left(x_N^{k+1}\right)}{dx_N^{k+1}}\frac{dx_N^{k+1}}{d\alpha^k} \tag{7}$$

or

$$\frac{dU\left(\mathbf{x}^{k+1}\right)}{d\alpha^k} = \left\langle \left[\nabla U\left(\mathbf{x}^{k+1}\right)\right]^T, \frac{d\mathbf{x}^{k+1}}{d\alpha^k} \right\rangle \tag{8}$$

However, from equation (3) it follows that

$$\frac{d\mathbf{x}^{k+1}}{d\alpha^k} = \mathbf{d}^{k+1} = -\nabla U\left(\mathbf{x}^k\right) \tag{9}$$

Substituting equation (9) into (8) and (6), it follows that for Steepest Descent (Figure 2)

$$\left\langle \left[\nabla U\left(\mathbf{x}^{k+1}\right)\right]^T, \nabla U\left(\mathbf{x}^k\right) \right\rangle = 0 \tag{10}$$

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 1).

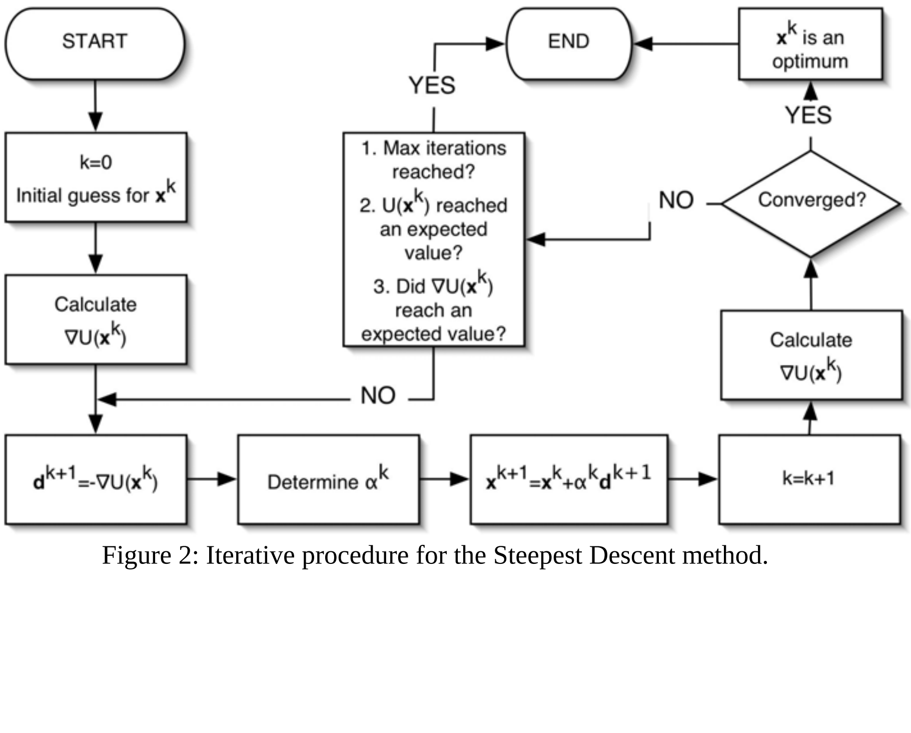

START
k=0
Initial guess for $\mathbf{x}^k$
Calculate
$\nabla U(\mathbf{x}^k)$
$\mathbf{d}^{k+1}=-\nabla U(\mathbf{x}^k)$
Determine α^k
$\mathbf{x}^{k+1}=\mathbf{x}^k+\alpha^k\mathbf{d}^{k+1}$
k=k+1
Calculate
$\nabla U(\mathbf{x}^k)$
Converged?
YES
$\mathbf{x}^k$ is an optimum
END
NO
1. Max iterations reached?
2. $U(\mathbf{x}^k)$ reached an expected value?
3. Did $\nabla U(\mathbf{x}^k)$ reach an expected value?
YES
NO

Figure 2: Iterative procedure for the Steepest Descent method.

In “real life” applications it is not possible to use equation (10) to evaluate the search step size. 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[2,3], Fibonacci search[2,3], golden search[2,3] and cubic interpolation[39], among others. However, for some realistic cases, the variation of the objective function with the search step size is not unimodal and then, more robust techniques are presented. The first one is the exhaustive search method and the second one is a technique based on exhaustive interpolation.

(a) Exhaustive Search[2,3]

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

The basic idea consists in uniformly dividing the domain that we are interested in (the initial uncertainty region), and finding the region where the maximum or minimum value are located. Let us call this domain I_0. Let us suppose, for instance, the situation shown in Figure 3, where an uncertainty interval I_0 was divided into eight sub regions, which are not necessarily the same size.

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

$$\begin{gathered} y_1 < y_2 < y_3 < y_4 < y_5 \\ y_5 > y_6 > y_7 > y_8 > y_9 \end{gathered} \tag{11}$$

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 a more refined grid could not only indicate this.

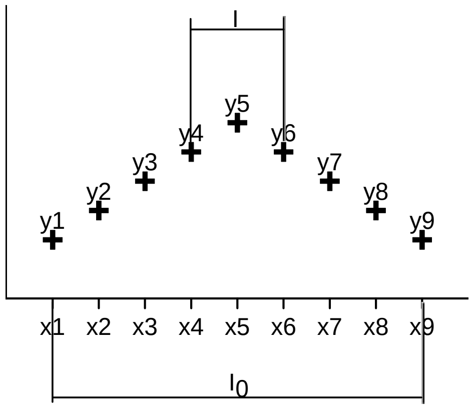


Figure 3: Exhaustive Search method.

Thus, the final uncertainty interval I is (x_6-x_4) and the optimum point is located somewhere inside this interval. It can be shown[2,3] that I is given by

$$I = \frac{2I_0}{n+1} \tag{12}$$

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 3, for $n = 9$, if we can assign the task of calculating the objective function at each point to an individual computer, the initial uncertainty region is reduced by 5 times within the time needed to just perform one calculation of the entire region using a single computer. Other more sophisticated methods, such as the Fibonacci method, for example, need sequential evaluations of the objective function. The Fibonacci method, for example, requires four objective function evaluations for the same reduction of the uncertainty region. Thus, in spite of its lack of efficiency in single processor applications, the Exhaustive Search method may be very efficient in parallel computing applications. A typical parallel computing arrangement is where one computer is the master and the other computers perform the evaluations of the objective function at each of the locations.

(b) Exhaustive Interpolation Search

This method is an improvement over the previous one, in that it requires fewer calculations to find the location of the minima. The method starts as the previous one, where domain is divided into several regions, where the objective functions are evaluated. The objective function is evaluated at a number of points in this domain. Next, a large number of points needs to be generated inside this domain and the objective function at these new points is estimated by spline fitting at the original points and interpolating at the new points using cubic splines[40], B-splines[41], kriging[42] or other interpolants. Interrogating these interpolated values we can find the region where the maximum or minimum values are located. The process can be repeated until a sufficiently small interval of uncertainty is obtained.

3.2. Conjugate Gradient Method[1-26]

The Steepest Descent method, in general, converges slowly for non-quadratic functions, since optimum search step sizes produce orthogonal gradients between two successive iterations. The Conjugate Gradient method tries to improve the convergence rate of the Steepest Descent method by choosing the directions of descent that reach the minimum value of the objective function faster. The iterative process for this method is given by the same equation used in the Steepest Descent method, equation (3). The difference is in the formulation for the direction of descent, which, for the Conjugate Gradient method, is given as a conjugation of the gradient and the direction of descent of the previous iteration

$$\mathbf{d}^{k+1} = -\nabla\left(\mathbf{x}^{k}\right) + \gamma^{k}\mathbf{d}^{k-1} + \psi^{k}\mathbf{d}^{q} \tag{13}$$

where γ^k and ψ^k are conjugation coefficients. The superscript q in equation (14) 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[15] 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 (14)[1,5,6,7-19]. In the Fletcher-Reeves version, the conjugation coefficients γ^k and ψ^k are obtained from the following expressions[1,5,6,9,11-16]

$$\gamma^k = \frac{\left\| \nabla U\left(\mathbf{x}^k\right) \right\|^2}{\left\| \nabla U\left(\mathbf{x}^{k-1}\right) \right\|^2}, \text{ with } \gamma^0 = 0 \text{ for } k=0 \tag{14.a}$$

$$\psi^k = 0, \text{ for } k = 0,1,2,\ldots \tag{14.b}$$

In the Polak-Ribiere version of the Conjugate Gradient method[1,6,8-10,15,18] the conjugation coefficients are given by

$$\gamma^k = \frac{[\nabla U\left(\mathbf{x}^k\right)]^T \left[\nabla U\left(\mathbf{x}^k\right) - \nabla U\left(\mathbf{x}^{k-1}\right)\right]}{\left\| \nabla U\left(\mathbf{x}^{k-1}\right) \right\|^2}, \text{ with } \gamma^0 = 0 \text{ for } k=0 \tag{15.a}$$

$$\psi^k = 0, \text{ for } k = 0,1,2,\ldots \tag{15.b}$$

Based on a previous work by Beale[19], Powell[15] suggested the following expressions for the conjugation coefficients, which gives the so-called Powell-Beale's version of the Conjugate Gradient method[9,15,19]

$$\gamma^k = \frac{[\nabla U(\mathbf{x}^k)]^T [\nabla U(\mathbf{x}^k) - \nabla U(\mathbf{x}^{k-1})]}{[\mathbf{d}^{k-1}]^T [\nabla U(\mathbf{x}^k) - \nabla U(\mathbf{x}^{k-1})]}, \text{ with } \gamma^0 = 0 \text{ for } k=0 \tag{16.a}$$

$$\psi^k = \frac{[\nabla U(\mathbf{x}^k)]^T [\nabla U(\mathbf{x}^{q+1}) - \nabla U(\mathbf{x}^q)]}{[\mathbf{d}^q]^T [\nabla U(\mathbf{x}^{q+1}) - \nabla U(\mathbf{x}^q)]}, \text{ with } \gamma^0 = 0 \text{ for } k=0 \tag{16.b}$$

In accordance with Powell[15], the application of the conjugate gradient method with the conjugation coefficients given by equations (16) requires restarting when gradients at successive iterations tend to be non-orthogonal (which is a measure of the local non-linearity of the problem) and when the direction of descent is not sufficiently downhill. Restarting is performed by making $\psi^k = 0$ in equation (14).

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

$$ABS\left([\nabla(\mathbf{x}^{k-1})]^T \nabla(\mathbf{x}^k)\right) \geq 0.2 \left\| \nabla\left(\mathbf{x}^k\right) \right\|^2 \tag{17.a}$$

where ABS (.) denotes the absolute value.

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

$$[\mathbf{d}^k]^T \nabla\left(\mathbf{x}^k\right) \leq -1.2 \left\| \nabla\left(\mathbf{x}^k\right) \right\|^2 \tag{17.b}$$

$$[\mathbf{d}^k]^T \nabla\left(\mathbf{x}^k\right) \geq -0.8 \left\| \nabla\left(\mathbf{x}^k\right) \right\|^2 \tag{17.c}$$

We note that the coefficients 0.2, 1.2 and 0.8 appearing in equations (17.a-c) are empirical determined and are the same values used by Powell[15].

In Powell-Beale's version of the conjugate gradient method, the direction of descent given by equation (13) is computed in accordance with the following algorithm for $k \geq 1$[15]:

STEP 1: Test the inequality (17.a). If it is true set $q = k$-1.

STEP 2: Compute γ^k with equation (16.a).

STEP 3: If $k = q+1$ set $\psi^k = 0$. If $k \neq q+1$ compute ψ^k with equation (16.b).
STEP 4: Compute the search direction $\mathbf{d}^{k+1}$ with equation (13).
STEP 5: If $k \neq q+1$ test the inequalities (17.b,c). If either one of them is satisfied set $q = k-1$ and $\psi^k=0$. Then recompute the search direction with equation (13).

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 (13). We note that the conjugation coefficients γ^k given by equations (14.a), (15.a) and (16.a) are equivalent for quadratic functions, because the gradients at different iterations are mutually orthogonal[1,15].

The same procedures used for the evaluation of the search step size in the Steepest Descent method can be employed here. Figure 4 illustrates the convergence history for the Fletcher-Reeves version of the Conjugate Gradient method for the same function presented in Figure 1. One can see that the Conjugate Gradient method is faster than the Steepest Descent. It is worth noting that the gradients between two successive iterations are no longer mutually orthogonal.

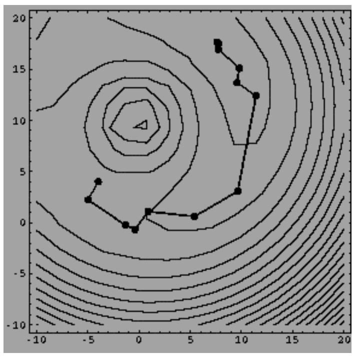


Figure 4: Convergence history for the Fletcher-Reeves version of the Conjugate Gradient method.

Colaço and Orlande[25] presented a comparison of Fletcher-Reeves’, Polak-Ribiere’s and Powell-Beale’s versions of the conjugate gradient method, as applied to the estimation of the heat transfer coefficient at the surface of a plate. This inverse problem was solved as a function estimation approach, by assuming that no information was available regarding the functional form of the unknown. Among the three versions tested for the Conjugate Gradient method, the method suggested by Powell and Beale appeared to be the best, as applied to the cases examined in that paper. This algorithm did not present the anomalous increase of the functional as observed with the other versions, and its average rates of reduction of the functional were the largest. As a result, generally, the smallest values for the RMS error of the estimated functions were obtained with Powell-Beale’s version of the conjugate gradient method.

Figure 5 shows the iterative procedure for the Fletcher-Reeves version of the Conjugate Gradient method.

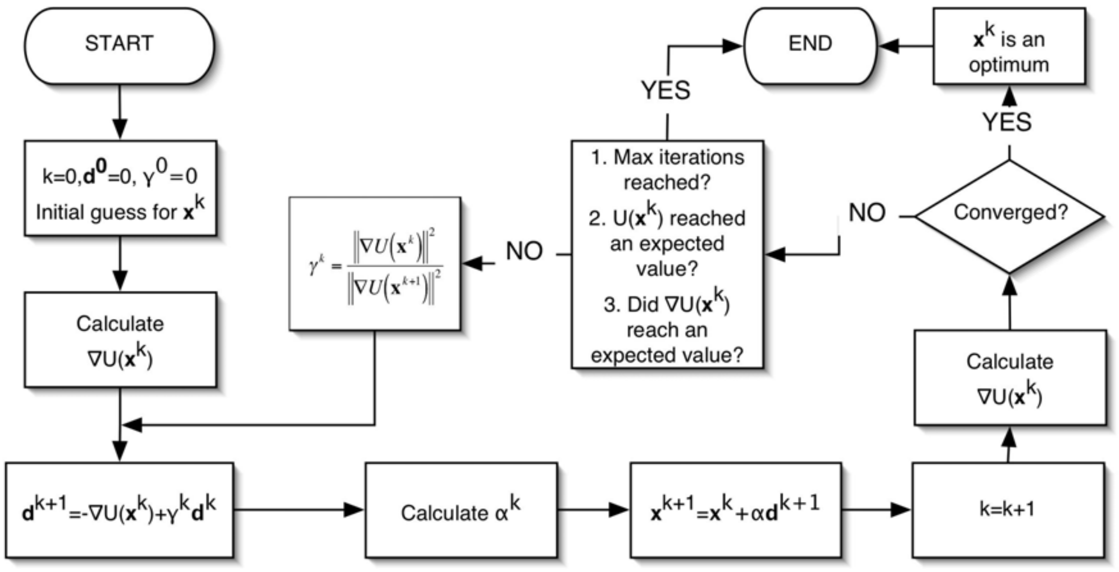


Figure 5: Iterative procedure for the Fletcher-Reeves version of the Conjugate Gradient method.

3.3. Newton-Raphson Method[1-4]

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

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

$$U(\mathbf{x}+\mathbf{h}) = U(\mathbf{x}) + \nabla U(\mathbf{x})^T \mathbf{h} + \frac{1}{2}\mathbf{h}^T \mathrm{D}^2 U(\mathbf{x})\mathbf{h} + O(\mathbf{h}^3) \tag{18}$$

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

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

$$\nabla U(\mathbf{x}+\mathbf{h}) \cong \nabla U(\mathbf{x}) + \mathrm{D}^2 U(\mathbf{x})\mathbf{h} \tag{19}$$

The optimum is obtained when the left side of equation (19) vanishes. Thus, we have

$$\mathbf{h}_{optimum} \cong -\left[\mathrm{D}^2 U(\mathbf{x})\right]^{-1} \nabla U(\mathbf{x}) \tag{20}$$

and the vector that optimizes the function $U(\mathbf{x})$ is

$$\left(\mathbf{x}+\mathbf{h}_{optimum}\right) \cong \mathbf{x} - \left[\mathrm{D}^2 U(\mathbf{x})\right]^{-1} \nabla U(\mathbf{x}) \tag{21}$$

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 (3) where the direction of descent is given by

$$\mathbf{d}^{k+1} = -\left[\mathrm{D}^2 U(\mathbf{x}^k)\right]^{-1} \nabla U(\mathbf{x}^k) \tag{22}$$

The Newton-Raphson method is faster than the Conjugate Gradient method as demonstrated in Figure 6. However, the calculation of the Hessian matrix coefficients takes a long time to evaluate. Figure 7 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.

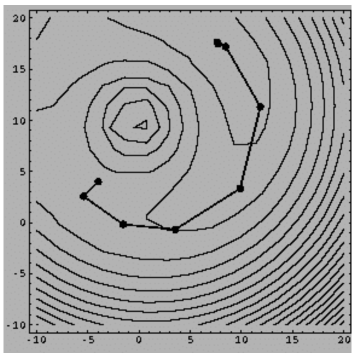
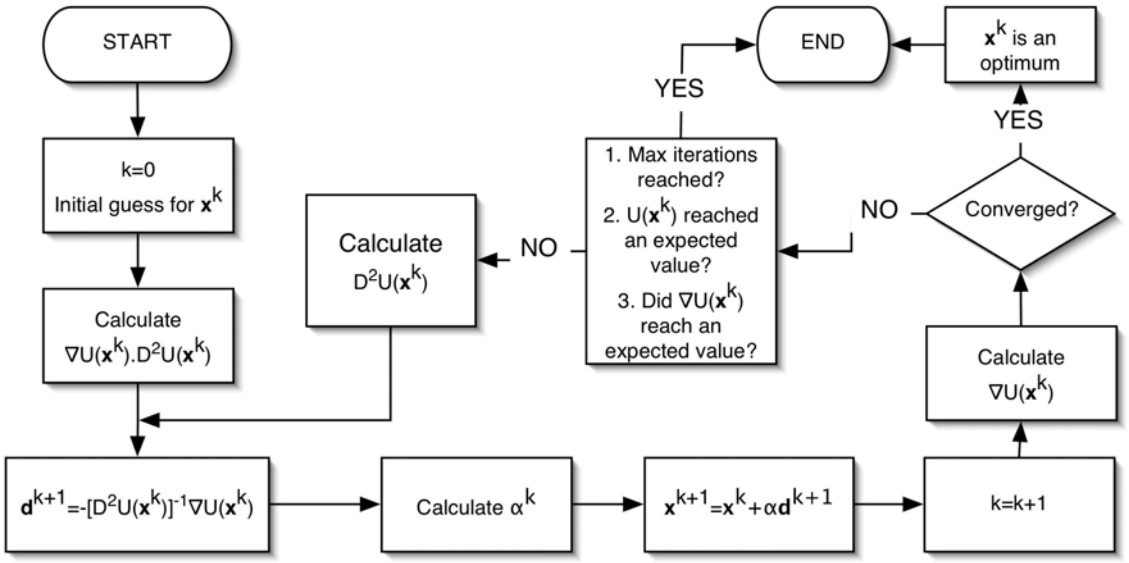


Figure 6: Convergence history for the Newton-Raphson method.


START
k=0
Initial guess for $\mathbf{x}^k$
Calculate
$\nabla U(\mathbf{x}^k).D^2U(\mathbf{x}^k)$
Calculate
$D^2U(\mathbf{x}^k)$
NO
1. Max iterations reached?
2. $U(\mathbf{x}^k)$ reached an expected value?
3. Did $\nabla U(\mathbf{x}^k)$ reach an expected value?
YES
END
$\mathbf{x}^k$ is an optimum
YES
NO
Converged?
Calculate
$\nabla U(\mathbf{x}^k)$
$\mathbf{d}^{k+1}=-[D^2U(\mathbf{x}^k)]^{-1}\nabla U(\mathbf{x}^k)$
Calculate α^k
$\mathbf{x}^{k+1}=\mathbf{x}^k+\alpha\mathbf{d}^{k+1}$
k=k+1
Figure 7: Iterative procedure for the basic Newton-Raphson method implementation.

3.4. Quasi Newton Methods[1-4]

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

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

$$\mathbf{H}^k = \left[D^2 U(\mathbf{x}^k)\right]^{-1} \tag{23}$$

Thus, the quasi Newton methods follow the general iterative procedure given by equation (4) where the direction of descent is given by

$$\mathbf{d}^{k+1} = -\mathbf{H}^k \nabla U\left(\mathbf{x}^k\right) \tag{24}$$

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

$$\mathbf{H}^{k} = \mathbf{H}^{k-1} + \mathbf{M}^{k-1} + \mathbf{N}^{k-1} \text{ for } k = 1,2,\ldots \tag{25.a}$$

$$\mathbf{H}^{k} = \mathbf{I} \text{ for } k = 0 \tag{25.b}$$

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

Different quasi-Newton methods can be found depending on the choice for the matrices **M** and **N**. For the Davidon-Fletcher-Powell (DFP) method[43,44], such matrices are given by

$$\mathbf{M}^{k-1} = \alpha^{k-1} \frac{\mathbf{d}^{k-1}\left(\mathbf{d}^{k-1}\right)^{T}}{\left(\mathbf{d}^{k-1}\right)^{T}\mathbf{Y}^{k-1}} \tag{26.a}$$

$$\mathbf{N}^{k-1} = -\frac{\left(\mathbf{H}^{k-1}\mathbf{Y}^{k-1}\right)\left(\mathbf{H}^{k-1}\mathbf{Y}^{k-1}\right)^{T}}{\left(\mathbf{Y}^{k-1}\right)^{T}\mathbf{H}^{k-1}\mathbf{Y}^{k-1}} \tag{26.b}$$

where

$$\mathbf{Y}^{k-1} = \nabla U\left(\mathbf{x}^{k}\right) - \nabla U\left(\mathbf{x}^{k-1}\right) \tag{26.c}$$

Figure 8 shows the results for the minimization of the objective function shown before, using the DFP method. One can see that its convergence rate is between the Conjugate Gradient method and the Newton-Raphson method.

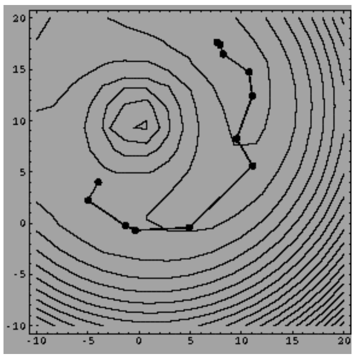


Figure 8: Convergence history for the DFP method.

Note that, since the matrix **H** is iteratively calculated, some errors can be propagated and, in general, the method needs to be restarted after certain number of iterations[39]. Also, since the matrix **M** depends on the choice of the search step size α, the method is very sensitive to its value. A variation of the DFP method is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method[45-48], which is less sensitive to the choice of the search step size. For this method, the matrices **M** and **N** are calculated as

$$\mathbf{M}^{k-1} = \left(\frac{1+\left(\mathbf{Y}^{k-1}\right)^{T}\mathbf{H}^{k-1}\mathbf{Y}^{k-1}}{\left(\mathbf{Y}^{k-1}\right)^{T}\mathbf{d}^{k-1}}\right)\frac{\mathbf{d}^{k-1}\left(\mathbf{d}^{k-1}\right)^{T}}{\left(\mathbf{d}^{k-1}\right)^{T}\mathbf{Y}^{k-1}} \tag{27.a}$$

$$\mathbf{N}^{k-1} = -\frac{\mathbf{d}^{k-1}\left(\mathbf{Y}^{k-1}\right)^{T}\mathbf{H}^{k-1} + \mathbf{H}^{k-1}\mathbf{Y}^{k-1}\left(\mathbf{d}^{k-1}\right)^{T}}{\left(\mathbf{Y}^{k-1}\right)^{T}\mathbf{d}^{k-1}} \tag{27.b}$$

Figure 9 shows the iterative procedure for the BFGS method.

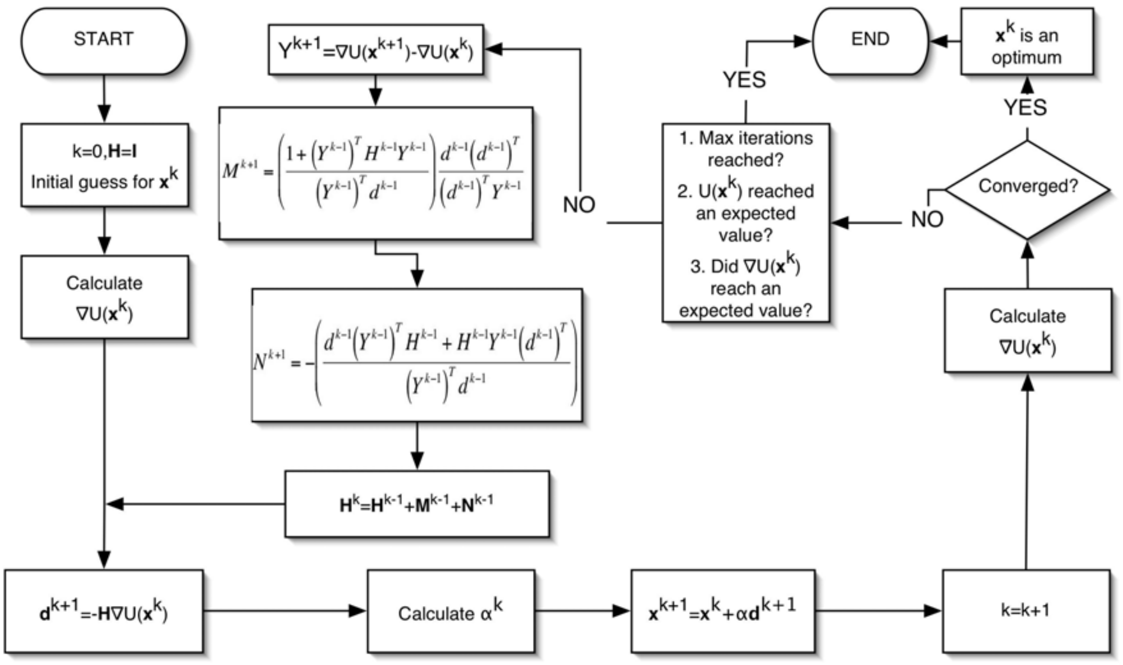

START
k=0,H=I
Initial guess for x^k
Calculate
∇U(x^k)
Y^{k+1}=∇U(x^{k+1})-∇U(x^k)
$M^{k+1} = \left(\frac{1+(Y^{k-1})^T H^{k-1} Y^{k-1}}{(Y^{k-1})^T d^{k-1}}\right)\frac{d^{k-1}(d^{k-1})^T}{(d^{k-1})^T Y^{k-1}}$
$N^{k+1} = -\left(\frac{d^{k-1}(Y^{k-1})^T H^{k-1} + H^{k-1} Y^{k-1}(d^{k-1})^T}{(Y^{k-1})^T d^{k-1}}\right)$
H^k=H^{k-1}+M^{k-1}+N^{k-1}
d^{k+1}=-H∇U(x^k)
Calculate α^k
x^{k+1}=x^k+αd^{k+1}
k=k+1
NO
1. Max iterations reached?
2. U(x^k) reached an expected value?
3. Did ∇U(x^k) reach an expected value?
YES
END
x^k is an optimum
YES
Converged?
NO
Calculate
∇U(x^k)

Figure 9: 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 (21). The search step size prevents the method from jumping to a maximum value when we look for a minimum and vice-versa. Figures 10 and 11 show the influence of the initial guess for all four methods for a Rosenbrock “banana-shape” function.

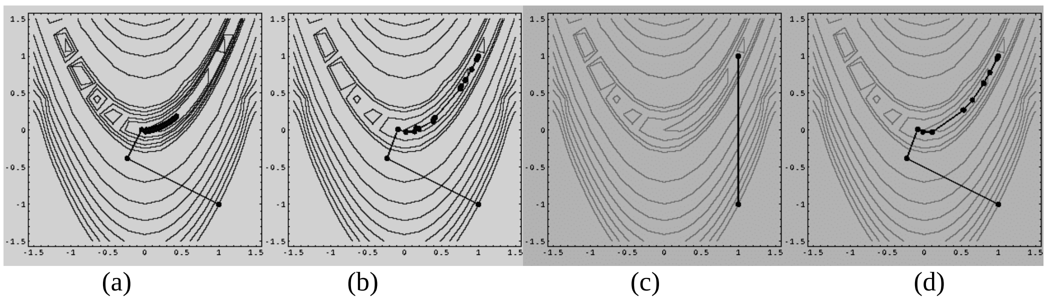

(a) (b) (c) (d)

Figure 10: First initial guess for the (a) Steepest Descent, (b) Conjugate Gradient, (c) Newton-Raphson and (d) DFP methods.

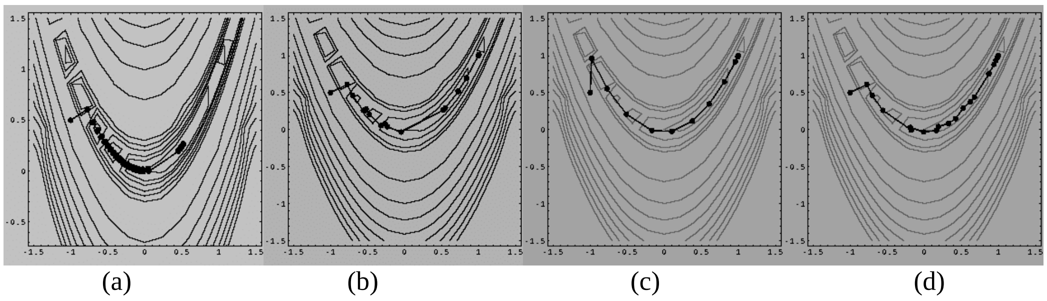

(a) (b) (c) (d)

Figure 11: Second initial guess for the (a) Steepest Descent, (b) Conjugate Gradient, (c) Newton-Raphson and (d) DFP methods.

There are other classes of powerful deterministic methods, as for example, the Levenberg-Marquardt method[49-50], which will not be discussed here. The reader is, however, encouraged to look at the references for a more comprehensive understanding of these methods.

4. EVOLUTIONARY AND STOCHASTIC METHODS

In this section methods like the Genetic Algorithms, Differential Evolution, Particle Swarm and Simulated Annealing will be addressed. Some practical aspects and limitations of such methods will be addressed. In section 5 we will combine this class of methods with the deterministic ones in so-called Hybrid Optimization methods.

Evolutionary methods, as opposite to the deterministic methods, do not rely, in general, on a strong mathematical basis. They attempt to mimic nature in its process of searching for the optimum.

4.1. Genetic Algorithms[32]

Genetic algorithms are heuristic global optimization methods that are based on the process of natural selection. Starting from a randomly generated population of 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[32,51].

The basic Genetic Algorithm works with a collection or population of potential 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 potential 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 know 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 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 object function.

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

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

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

4.2. Differential Evolution[52]

The Differential Evolution method is an evolutionary method based on Darwin's theory of evolution of the species[33]. This non-gradient based optimization method was created in 1995[52] 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

$$\mathbf{x}_i^{k+1} = \delta_1 \mathbf{x}_i^k + \delta_2 \left[\boldsymbol{\alpha} + F\left(\boldsymbol{\beta} - \boldsymbol{\gamma} \right) \right] \tag{28}$$

where

$\mathbf{x}_\mathrm{i}$ is the *i*-th individual of the vector of parameters.

α, β and γ are three members of population matrix **P**, randomly choosen.

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

k is a counter for the generations.

δ_1 and δ_2 delta Dirac functions that define the mutation.

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

The binomial crossover is given as

$$\delta_1 = \begin{matrix} 0, & \text{if} \quad R < CR \\ 1, & \text{if} \quad R > CR \end{matrix} \qquad \delta_2 = \begin{matrix} 1, & \text{if} \quad R < CR \\ 0, & \text{if} \quad R > CR \end{matrix} \tag{29.a,b}$$

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

Figure 12 shows the iterative procedure for the Differential Evolution method.

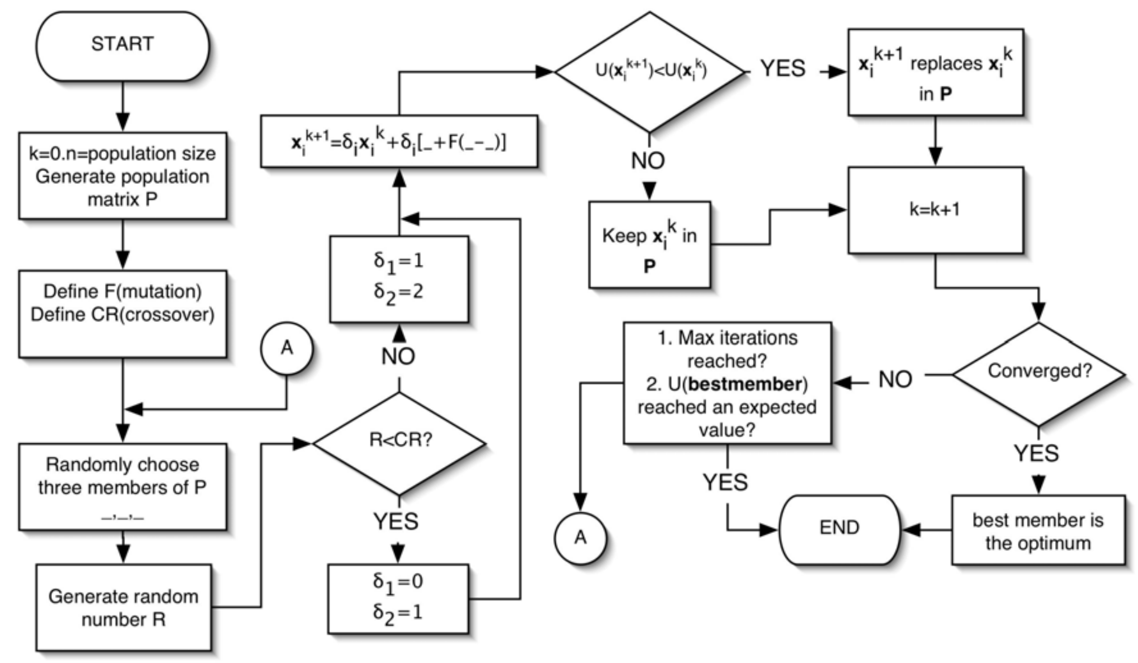

START
k=0.n=population size
Generate population matrix P
Define F(mutation)
Define CR(crossover)
A
Randomly choose three members of P
Generate random number R
R<CR?
YES
NO
δ1=0 δ2=1
δ1=1 δ2=2
$x_i^{k+1}=δ_i x_i^k+δ_i[\_+F(\_-\_)]$
$U(x_i^{k+1})<U(x_i^k)$
YES
NO
x_i^{k+1} replaces x_i^k in P
Keep x_i^k in P
k=k+1
Converged?
NO
YES
1. Max iterations reached?
2. U(bestmember) reached an expected value?
YES
END
best member is the optimum
A

Figure 12: Iterative procedure for the Differential Evolution method.

4.3. Particle Swarm[27-30]

This non-gradient based optimization method was created in 1995 by an electrical engineer (Russel Eberhart) and a social psychologist (James Kennedy)[27-30] 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

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^{k+1} \tag{30.a}$$

$$\mathbf{v}_i^{k+1} = \alpha\mathbf{v}_i^k + \beta\mathbf{r}_{1i}\left(\mathbf{p}_i - \mathbf{x}_i^k\right) + \beta\mathbf{r}_{2i}\left(\mathbf{p}_g - \mathbf{x}_i^k\right) \tag{30.b}$$

where:

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

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

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

$\mathbf{p}_i$ is the best value found for the vector $\mathbf{x}_i$.

$\mathbf{p}_g$ is the best value found for the entire population.

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

In equation (30.b), the second term on the right hand side represents the individuality and the third term the sociability. The first term on the right-hand side represents the inertia of the particles and, in general, must be decreased as the iterative process proceeds. In this equation, the vector $\mathbf{p}_i$ represents the best value ever found for the i-th component vector of parameters $\mathbf{x}_i$ during the iterative process. Thus, the individuality term involves the comparison between the current value of the i-th individual $\mathbf{x}_i$ and its best value in the past. The vector $\mathbf{p}_g$ is the best value ever found for the entire population of parameters (not only the i-th individual). Thus the sociability term compares $\mathbf{x}_i$ with the best value of the entire population in the past.

Figure 13 shows the iterative procedure for the Particle Swarm method.

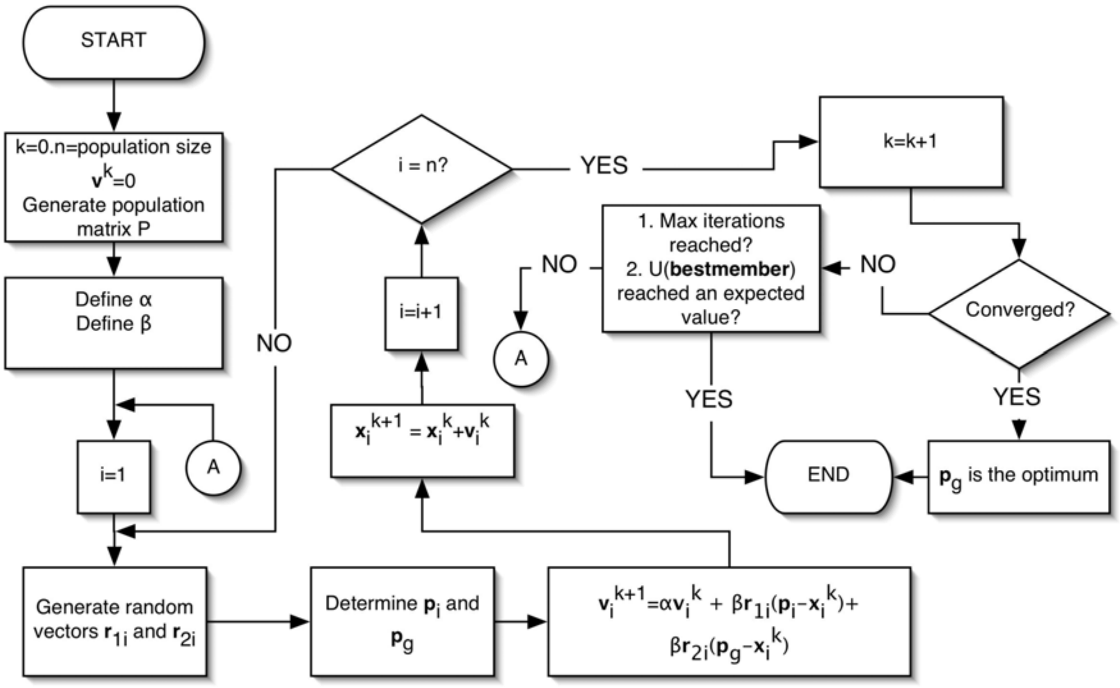

START
k=0.n=population size
$\mathbf{v}^k=0$
Generate population matrix P
Define α
Define β
i=1
A
NO
Generate random vectors $\mathbf{r}_{1i}$ and $\mathbf{r}_{2i}$
Determine $\mathbf{p}_i$ and $\mathbf{p}_g$
$\mathbf{v}_i^{k+1}=\alpha\mathbf{v}_i^k+\beta\mathbf{r}_{1i}(\mathbf{p}_i-\mathbf{x}_i^k)+\beta\mathbf{r}_{2i}(\mathbf{p}_g-\mathbf{x}_i^k)$
$\mathbf{x}_i^{k+1}=\mathbf{x}_i^k+\mathbf{v}_i^k$
i=i+1
i = n?
YES
k=k+1
Converged?
NO
1. Max iterations reached?
2. U(**bestmember**) reached an expected value?
NO
A
YES
END
YES
$\mathbf{p}_g$ is the optimum

Figure 13: Iterative procedure for the Particle Swarm method.

4.4. Simulated Annealing[53,54]

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

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

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

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

$$\mathrm{Prob}(E) \quad \propto \quad e^{(-E/KT)} \tag{31}$$

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 (31) tells us that even at low temperatures, there is a chance, although small, that the system is at a high energy level. 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 14 shows the iterative procedure for the Simulated Annealing method. The procedure starts generating a population of individuals of the same size of the number of variables ($n = m$), in such a way that the population matrix is a square matrix. Then, the initial temperature (T), the reducing ratio (RT), the number of cycles (N_s) and the number of iterations of the annealing process (N_{it}) are selected. After N_s*n function evaluations, each element of the step length V is adjusted so that approximately half of all function evaluations are accepted. The suggested value for the number of cycles is 20. After $N_{it}*N_s*n$ function evaluations, the temperature (T) is changed by the factor RT. The value suggested for the number of iterations by Corana et al. [53] is MAX(100, 5*n).

The iterative process follows the following equation:

$$x_i^1 = x_i^0 + RV_i \tag{32}$$

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

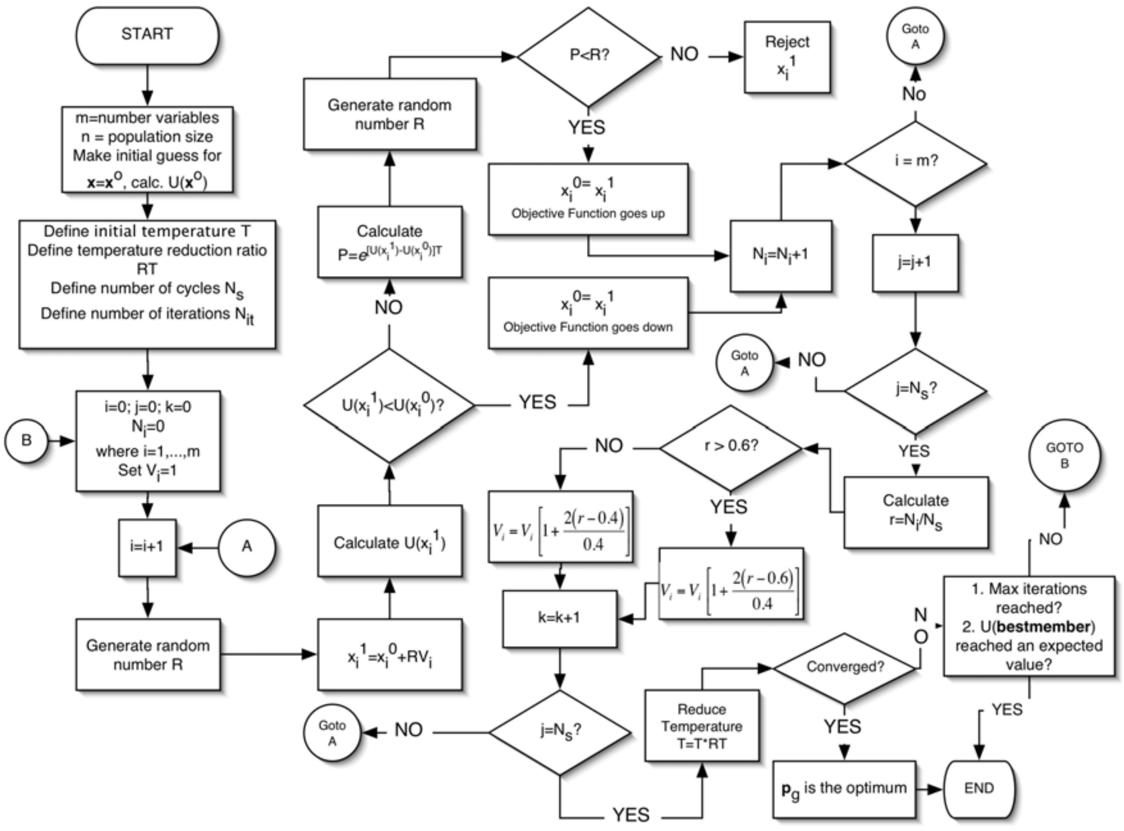


Figure 14: Iterative procedure for the Simulated Annealing method.

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

$$P = e^{\left[U\left(x_i^1\right) - U\left(x_i^0\right)\right]/T} \tag{33}$$

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 instead 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

$$T(i+1) = RT * T(i) \tag{34}$$

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

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

5. HYBRID OPTIMIZATION METHODS[34-38]

The Hybrid Optimization methods are not more than a combination of the deterministic and the evolutionary/stochastic methods, in the sense that they try to use the advantages of each of these methods. The Hybrid Optimization method usually employs an evolutionary/stochastic method to locate a region where the global extreme point is located and then automatically switches to a deterministic method to get to the exact point faster. The Hybrid Optimization method proposed here is quite simple conceptually, although its computational implementation is more involved. The global procedure is illustrated in Figure 15. The driven module is very often the Particle Swarm method, which often performs most of the optimization task. When certain percent of the particles find a minima (let us say, some birds already found their best nesting place), the algorithm switches automatically to the Differential Evolution method and the particles (birds) are forced to breed. If there is an improvement in the objective function, the algorithm returns to the Particle Swarm method, meaning that some other region is more prone to having a global minimum. If there is no improvement on the objective function, this can indicate that this region already contains the global value expected and the algorithm automatically switches to the BFGS method in order to find its location more precisely. In Figure 15, the algorithm returns to the Particle Swarm method in order to check if there are no changes in this location and the entire procedure repeats itself. After some maximum number of iterations is performed (e.g., five) the process stops.

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

Notice that this Hybrid Optimization method differs considerably from the earlier version that performed automatic switching among six classical optimization modules[55].

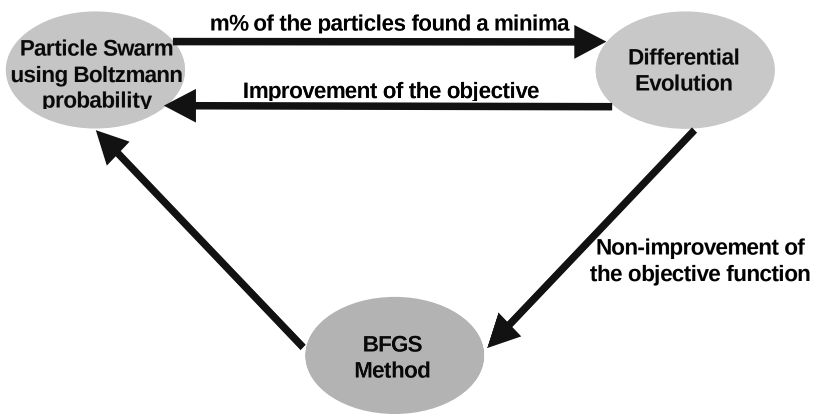

Particle Swarm using Boltzmann probability
m% of the particles found a minima
Differential Evolution
Improvement of the objective
Non-improvement of the objective function
BFGS Method

Figure 15: Global procedure for the Hybrid Optimization method.

6. COMPARISONS AMONG THE DETERMINISTIC, EVOLUTIONARY AND HYBRID METHODS

Since the evolutionary and the stochastic methods use random numbers in their formulations, each optimization with such a method leads to slightly different results. However, this section is included to give the reader a brief insight in the basic performance differences among various methods. The functions to be analyzed do not correspond to a physical model. Rather, they are functions constructed in such a way that they have several local minima, making the optimization process more challenging. In all test cases presented the maximum allowed number of iterations was very large. The BFGS method stopped when the gradient of the objective function changed its sign, meaning that a local or global minimum was found.

(a) Minimization of the Griewangk's function[43]

For this first case, we will show the performance of several of the optimizers to find the optimum of the Griewangk's function, which is defined as

$$U = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad x \in \left]-600,600\right[\tag{35}$$

The global minima for this function is located at $\mathbf{x} = 0$ and is $U(\mathbf{x}) = 0$. For a two-dimensional test case, such function is shown in Figure 16 in three levels of local resolution. One can see that this function has an extremely large number of local minima, making the optimization task of finding the global minimum quite difficult.

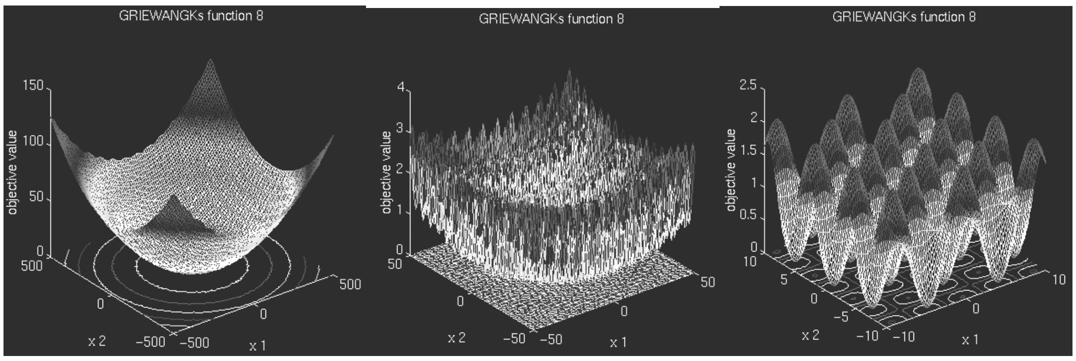


Figure 16: Griewangk's function.

Figure 17 shows the results for the optimization task using the (a) BFGS, (b) Differential Evolution, (c) Simulated Annealing, (d) Particle Swarm and (e) Hybrid Optimization methods. One can see that the evolutionary/stochastic methods are a little bit better than the BFGS method. However, only the Hybrid Optimization method is capable of locating the global optimum value of this function.

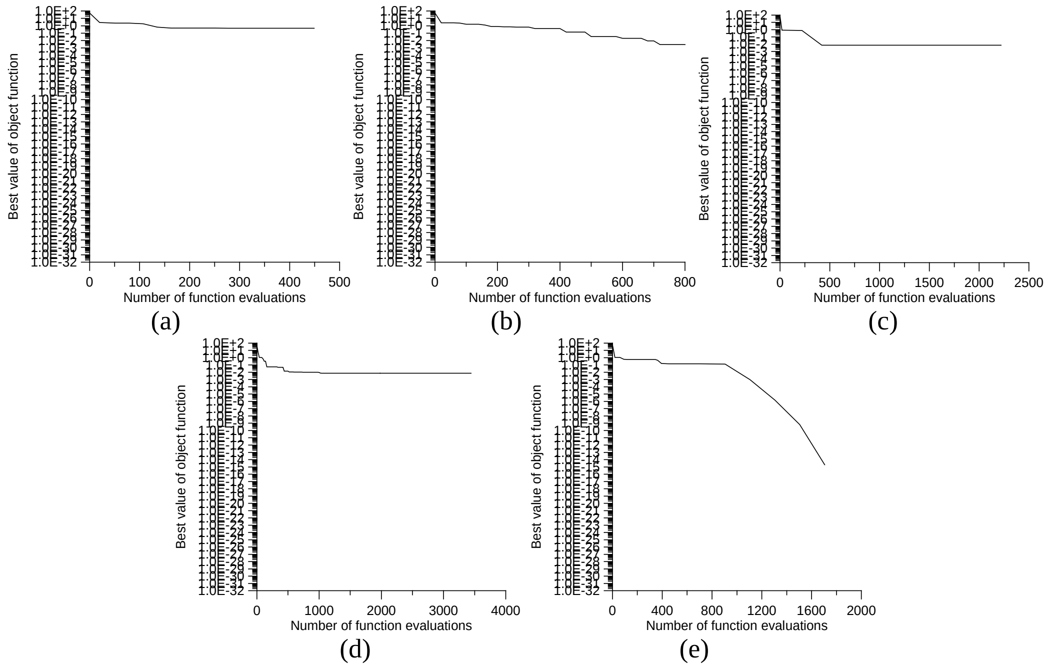


Figure 17: Comparison of the optimizers for the Griewangk's function.

(b) Minimization of the Levy function[43]

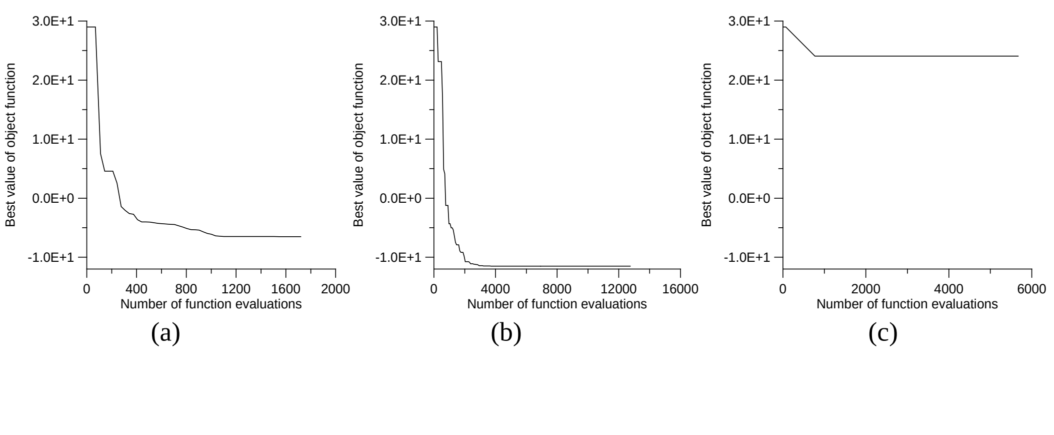
The Levy function, which is defined as

$$U = \sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 \left(1 + \sin^2(3\pi x_{i+1})\right) + (x_n - 1)\left(1 + \sin^2(2\pi x_n)\right) \tag{36}$$

$$x \in \;]-5,5[$$

will now be analyzed for a test case with 7 variables. The global minima for this function is located at **x** = (1,1,1,1,1,1,-4.754402) and its value is *U*(**x**) = -11.

Figure 18 shows that the Simulated Annealing method is the worst method in this case with the BFGS method performing only slightly better. The Differential Evolution, Particle Swarm and Hybrid Optimization methods were capable of minimizing this function. However, the last two methods were faster, while the Hybrid Optimization method was the fastest.



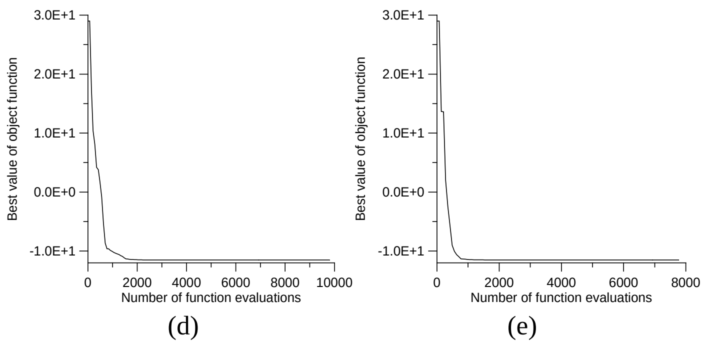


Figure 18: Comparison of the optimizers for the Levy function using the (a) BFGS, (b) Differential Evolution, (c) Simulated Annealing, (d) Particle Swarm and (e) Hybrid Optimization method.

(c) Minimization of the Rosen's function[43]

For this case, we will show the performance of several of the optimizers to find the optimum of the Rosen's function, which is defined as

$$U = 0.25x_1^4 - 3x_1^3 + 11x_1^2 - 13x_1 + 0.25x_2^4 - 3x_2^3 + 11x_2^2 - 13x_2 \qquad (37)$$
$$x \in \;]0,6[$$

The global minima for this function is located at **x** = (5.33006,5.33006) and its value is U(**x**) = -18.568. For a two-dimensional test case, such function is shown in Figure 19. One can see that the function has four minimums and one maximum.

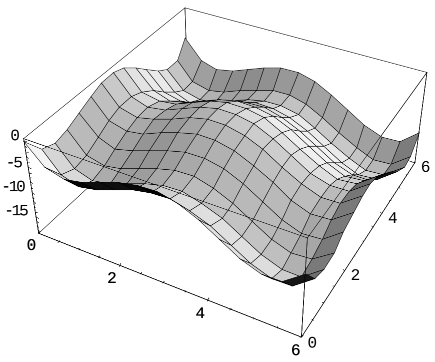
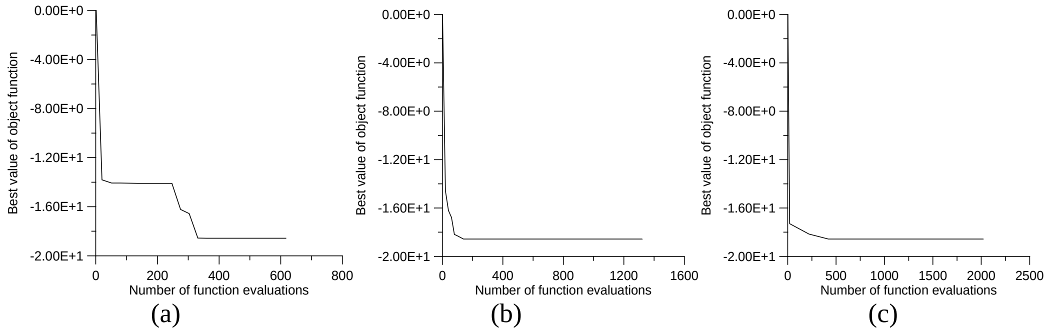
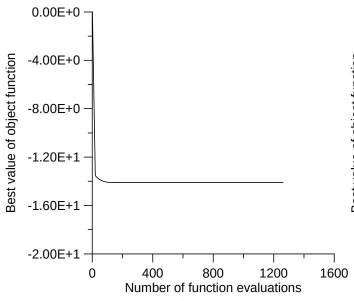
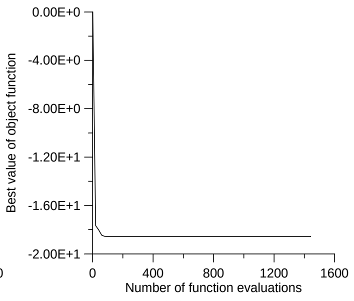


Figure 19: Rosen's function.

Figure 20 shows that all methods, except the Particle Swarm were capable of minimizing this relatively simple function. The faster ones were the Differential Evolution and the Hybrid Optimization methods with the last one a little bit faster.







(d) (e)

Figure 20: Comparison of the optimizers for the Rosen's function using the (a) BFGS, (b) Differential Evolution, (c) Simulated Annealing, (d) Particle Swarm and (e) Hybrid Optimization method.

(d) Minimization of the Schwefel's function[43]

For this first case, we will show the performance of several of the optimizers to find the optimum of the Schwefel's function, which is defined as

$$U = \sum_{i=1}^{n} - x_i \sin\left(\sqrt{abs(x_i)}\right) \tag{38}$$

$$x \in]-500,500[$$

The global minima for this function is located at **x** = 420.9687 and its value is $U(\mathbf{x})$ = -n*418.9829, where n is the number of variables. For a two-dimensional test case, such function is shown in Figure 21. One can see that the function has an extreme number of local minima, making the optimization task quite difficult.

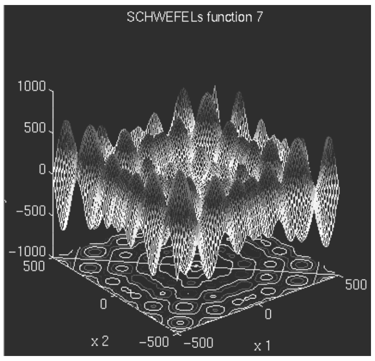
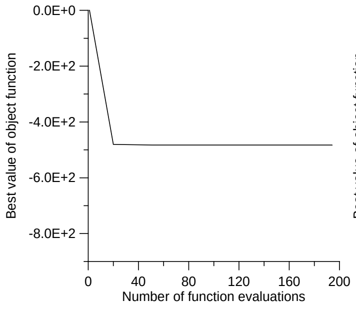
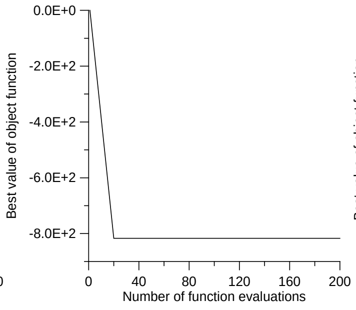
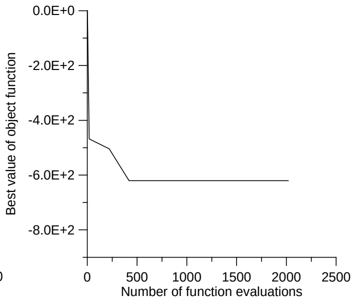


Figure 21: Schwefel's function

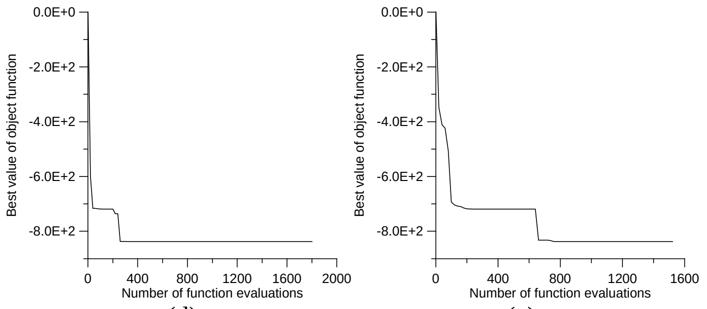
Figure 22 shows that only the Particle Swarm and the Hybrid Optimization methods were capable of fully minimizing this function, with the former been a little bit faster.







(a) (b) (c)



(d) (e)

Figure 22: Comparison of the optimizers for the Schwefel's function using the (a) BFGS, (b) Differential Evolution, (c) Simulated Annealing, (d) Particle Swarm and (e) Hybrid Optimization method.

(d) Minimization of the Schubert's function[43]

For this first case, we will show the performance of several of the optimizers to find the optimum of the Schubert's function, which is defined as

$$U = -\sum_{i=1}^{n}\sum_{j=1}^{5} j\sin\left(\left(j+1\right)x_i + j\right) \tag{39}$$

$$x \in \;]-10,10[$$

The global minima for this function is $U(\mathbf{x})$ = -24.062499, For a two-dimensional test case, such function is shown in Figure 23. One can see that the function has a large number of local minima, making the optimization task quite difficult.

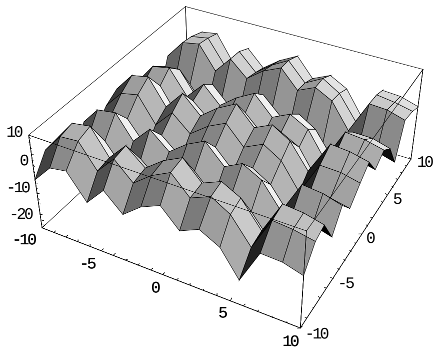
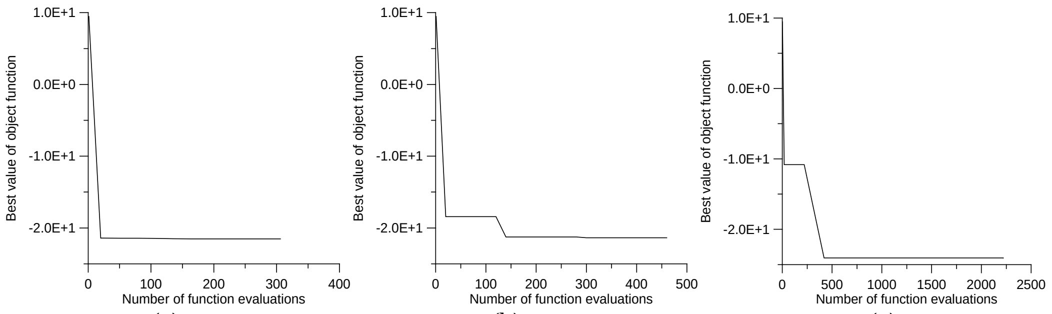
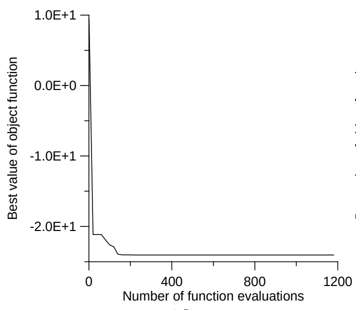


Figure 23: Schubert's function.

Figure 24 shows that the BFGS and the Differential Evolution methods were not able to fully minimize this function. Among the other methods, the Hybrid Optimization was the fastest.



(a) (b) (c)


(d)

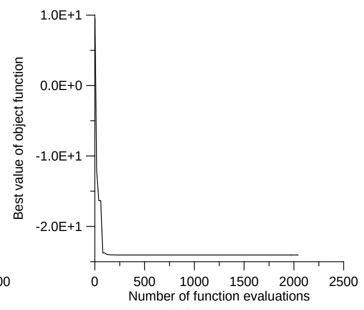

(e)

Figure 24: Comparison of the optimizers for the Schubert's function using the (a) BFGS, (b) Differential Evolution, (c) Simulated Annealing, (d) Particle Swarm and (e) Hybrid Optimization method.

From this brief comparison among the methods for various mathematical functions, we could see that the Hybrid Optimization method performed better in most of the test cases. It can reach the global minimum when other methods fail and it can reach it faster when the other methods work. For the next section we will show the application for some engineering test cases.

7. AN ALTERNATIVE HYBRID OPTIMIZER

In these examples, a slightly different Hybrid Optimization method was used[55], which had the following optimization modules; the Davidon-Fletcher-Powell (DFP) Gradient method, a Genetic Algorithm (GA), the Nelder-Mead (NM) Simplex method[44], quasi-Newton algorithm of Pshenichny-Danilin (LM)[56], Differential Evolution (DE), and Sequential Quadratic Programming (SQP)[57]. This Hybrid Optimization method is an older version of the one demonstrated previously in this Chapter, where some of the modules where discarded, due to their poor performance. Figure 25 shows a global procedure used for this older Hybrid Optimization method. In general, the Hybrid Optimization method described above has a performance equal or superior to this one.

This older hybrid optimization method[55] was successfully applied to problems involving the estimation of the diffusion coefficient and source terms[38] as well as problems involving magnetohydrodynamics[35-37] and electrohydrodynamics[34].

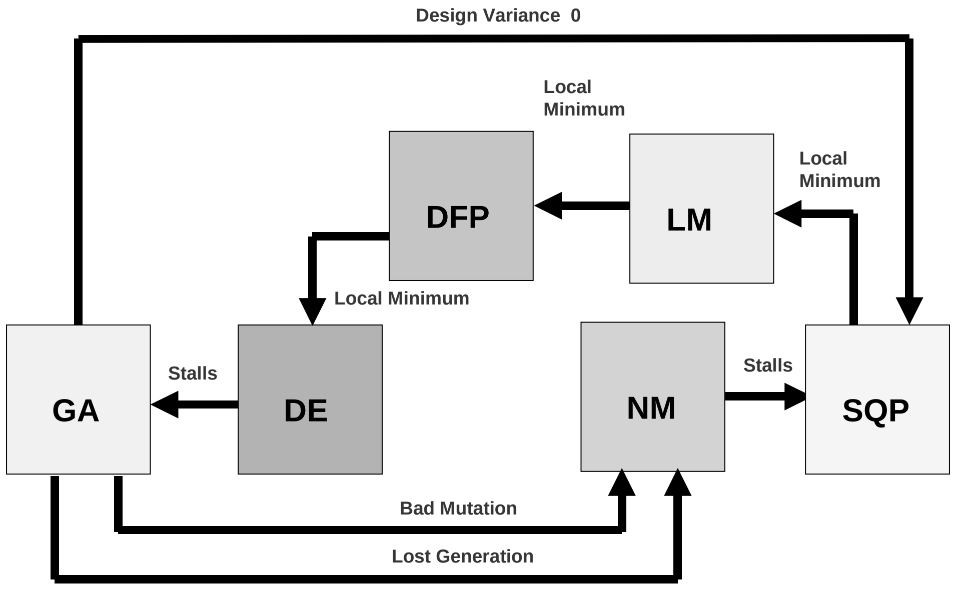


Figure 25: Global procedure for the older Hybrid Optimization method[55]

8. REFERENCES

[1] J. W. Daniel, *The Approximate Minimization of Functionals*, Prentice-Hall, Englewood Cliffs, (1971).

[2] Y. Jaluria, *Design and Optimization of Thermal Systems*, McGraw Hill, 1998.

[3] W. F. Stoecker, *Design of Thermal Systems*, McGraw Hill, (1989).

[4] A. D. Belegundu and T. R. Chandrupatla, *Optimization Concepts and Applications In Engineering*, Prentice Hall, (1999).

[5] O. M. Alifanov, "Solution of an Inverse Problem of Heat Conduction by Iteration Methods", *Journal of Engineering Physics*, **26** (4) 471-475 (1974).

[6] O. M. Alifanov, *Inverse Heat Transfer Problems*, Springer-Verlag, New York, (1994).

[7] E. A. Artyukhin, "Iterative Algorithms for Estimating Temperature-Dependent Thermophysical Characteristics", *1st Int. Conf. on Inverse Problems in Engineering: Theory and Practice*, pp.101-108., N. Zabaras, K. Woodburry and M. Raynaud (editors), Palm Coast, FL, June (1993).

[8] Y. Jarny, M. N. Ozisik and J. P. Bardon, "A General Optimization Method Using Adjoint Equation for Solving Multidimensional Inverse Heat Conduction", *Int. J. Heat Mass Transfer*, **34** (11) 2911-2919, (1991).

[9] B. Truffart, J. Jarny and D. Delaunay, "A General Optimization Algorithm to Solve 2-D Boundary Inverse Heat Conduction Problems Using Finite Elements", *1st Int. Conf. on Inverse Problems in Engineering: Theory and Practice*, pp.53-60, N. Zabaras, K. Woodburry and M. Raynaud (editors), Palm Coast, FL, June (1993).

[10] L. B. Dantas and H. R. B. Orlande, "A Function Estimation Approach for Determining Temperature-Dependent Thermophysical Properties", *Inverse Problems in Engineering*, **3**, 261-279, (1996).

[11] H. A. Machado and H. R. B. Orlande, "Inverse Analysis for Estimating the Timewise and Spacewise Variation of the Wall Heat Flux in a Parallel Plate Channel", *International Journal for Numerical Methods for Heat & Fluid Flow*, **7**, (7) 696-710, (1997).

[12] H. R. B. Orlande, M. J. Colaço and A. A. Malta, "Estimation of the Heat Transfer Coefficient in the Spray Cooling of Continuously Cast Slabs", *National Heat Transfer Conference*, ASME HTD-340, vol. 2, G. S. Dulikravich and K. A. Woodbury (editors), pp. 109-116, June (1997).

[13] C. H. Huang and C. H. Tsai, "A Shape Identification Problem in Estimating Time-Dependent Irregular Boundary Configurations", *National Heat Transfer Conference*, ASME HTD-340, vol. 2, G. S. Dulikravich and K. A. Woodbury (editors), pp. 41-48, (1997).

[14] J. P. Alencar Jr., H. R. B. Orlande and M. N. Ozisik, "A Generalized Coordinates Approach for the Solution of Inverse Heat Conduction Problems", *11th International Heat Transfer Conference*, Korea, **7**, 53-58, August (1998).

[15] M. J. D. Powell, "Restart Procedures for the Conjugate Gradient Method", *Mathematical Programming*, **12**, 241-254, (1977).

[16] R. Fletcher and C. M. Reeves, "Function Minimization by Conjugate Gradients", *Comput. Journal*, **7**, 149-154, (1964).

[17] M. R. Hestenes and E. Stiefel, "Method of Conjugate Gradients for Solving Linear Systems", *J. Res. Nat. Bur. Standards Sect. B*, **49**, 409-436, (1952).

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

[19] E. M. L. Beale, "A Derivation of Conjugate Gradients", *Numerical Methods for*

Nonlinear Optimization, F. A. Lootsma (editor), 39-43, (1972).

[20] M. J. Colaço and H. R. B. Orlande, "Inverse Forced Convection Problem of Simultaneous Estimation of Two Boundary Heat Fluxes in Irregularly Shaped Channels", *Numerical Heat Transfer, Part A*, **39**, 737-760, (2001).

[21] M. J. Colaço and H. R. B. Orlande, "Inverse Problem of Simultaneous Estimation of Two Boundary Heat Fluxes in Parallel Plate Channels", *Journal of the Brazilian Society of Mechanical Sciences,* vol. XXIII, (2), 201-215, (2001).

[22] M. J. Colaço and H. R. B. Orlande, "A Natural Convection Inverse Problem of Simultaneous Identification of Two Boundary Heat Fluxes in Rectangular Cavities", In: 12th International Heat Transfer Conf., Grenoble, France, (2002).

[23] M. J. Colaço and H. R. B. Orlande, "Inverse Natural Convection Problem of Simultaneous Estimation of Two Boundary Heat Fluxes in Irregular Cavities", International Journal of Heat and Mass Transfer, **49**, 1201-1215, (2003).

[24] M. J. Colaço and H. R. B. Orlande, "Estimation of the Heat Transfer Coefficient at the Surface of a Plate by Using the Conjugate Gradient Method", In: VII Encontro Nacional de Ciências Térmicas, Rio de Janeiro, Brazil, **1**, 189-194, (1998).

[25] M. J. Colaço and H. R. B. Orlande, "A Comparison of Different Versions of the Conjugate Gradient Method for Function Estimation", *Numerical Heat Transfer, Part A,* **36**, (2) 229-249, (1999).

[26] M. J. Colaço and H. R. B. Orlande, "A Function Estimation Approach for the Identification of the Transient Inlet Profile in Parallel Plate Channels", In: International Symposium on Inverse Problems in Engineering Mechanics, M. Tanaka and G. S. Dulikravich (editors), pp. 409-418, Nagano City, Japan, (2000).

[27] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization", *Proceedings of the 1995 IEEE International Conf. on Neural Networks*, **4**, 1942-1948, (1995).

[28] J. Kennedy, "Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance", *Proceedings of the 1999 Congress of Evolutionary Computation*, IEEE Press, **3**, 1931-1938, (1999).

[29] S. Naka, T. G. Yura and T. Fukuyama, "Practical Distribution State Estimation using Hybrid Particle Swarm Optimisation", *Proceedings IEEE Power Engineering Society Winter Meeting*, Columbus, OH, January 28-February 1, (2001).

[30] R. Eberhart, Y. Shi and J. Kennedy, *Swarm Intelligence,* Morgan Kaufmann, (2001).

[31] M. Dorigo and T. Stützle, *Ant Colony Optimization*, MIT Press, (2004).

[32] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning,* Addison-Wesley, (1989).

[33] C. Darwin, *On the Origin of Species*, John Murray, London, (1859).

[34] M. J. Colaço, G. S. Dulikravich and T. J. Martin, "Optimization of Wall Electrodes for Electro-Hydrodynamic Control of Natural Convection Effects During Solidification", *Materials & Manufacturing Processes*, **19**, 4, 719-736, (2004).

[35] M. J. Colaço, G. S. Dulikravich and T. J. Martin, "Reducing Convection Effects in Solidification by Appling Magnetic Fields Having Optimized Intensity Distribution", *ASME paper HT2003-47308, ASME Summer Heat Transfer Conferen*ce**,** Las Vegas, NV, July 21-23, (2003).

[36] G. S. Dulikravich, M. J. Colaço, T. J. Martin and S. Lee, "An Inverse Method Allowing User-Specified Layout of Magnetized Micro-Fibers in Solidifying Composites", *Journal of Composite Materials*, **37**, 15, 1351-1365, (2003).

[37] G. S. Dulikravich, M. J. Colaço, B. H. Dennis, T. J. Martin and S. Lee, "Optimization of Intensities, and Orientations of Magnets Controlling Melt Flow During Solidification", *Materials & Manufact. Processes*, **19**, 4, 695-718 (2004).

[38] M. J. Colaço, H. R. B. Orlande, G. S. Dulikravich and F. A. Rodrigues, "A Comparison of Two Solution Techniques for the Inverse Problem of Simultaneously Estimating the Spatial Variations of Diffusion", *ASME paper IMECE2003-42058, ASME IMECE 2003*, Washington, DC, Nov. 16-21, (2003).

[39] R. L. Fox, *Optimization Methods for Engineering Design*, Addison-Wesley Publishing Company, (1971).

[40] G. S. Dulikravich and T. J. Martin, "Inverse Design of Super-Elliptic Cooling Passages in Coated Turbine Blade Airfoils", *AIAA Journal of Thermophysics and Heat Transfer*, **8**, (2) 288-294, April-June, (1994).

[41] C. de Boor, *A Practical Guide to Splines*, Springer-Verlag, New York, (1978).

[42] M. A. Oliver and R. Webster, "Kriging: A Method of Interpolation for Geographical Information System", *Int. J. Geographical Information Systems*, **4**, (3), 313-332, (1990).

[43] GEATbx: *Genetic and Evolutionary Algorithm Toolbox for use with MATLAB, Version 1.91*, July, (1997).

[44] R. Fletcher, *Practical Methods of Optimization*, John Wiley & Sons, (2000).

[45] W. C. Davidon, *Variable Metric Method for Minimization*, Argonne Natl. Lab., ANL-5990, (1959).

[46] R. Fletcher and M. J. D. Powell, "A Rapidly Convergent Descent Method for Minimization", *Computer J.,* **6**, 163-168, (1963).

[47] C. G. Broyden, "A Class of Methods for Solving Nonlinear Simultaneous Equations", *Math. Comp*., **19**, 577-593, (1965).

[48] C. G. Broyden, "Quasi-Newton Methods and Their Applications to Function Minimization", *Math. Comp.*, **21**, 368-380, (1967).

[49] P. R. Gill, W. Murray and M. H. Wright, "The Levenberg-Marquardt Method", §4.7.3 in *Practical Optimization*, London, Academic Press, 136-137, (1981).

[50] M. N. Özisik and H. R. B. Orlande, *Inverse Heat Transfer: Fundamentals and Applications*, Taylor & Francis Inc, (2000).

[51] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, (2002).

[52] R. Storn and K. V. Price, "Minimizing the Real Function of the ICEC'96 Contest by Differential Evolution", *IEEE Conf. on Evolutionary Comput.*, 842-844, (1996).

[53] A. Corana, M. Marchesi, C. Martini and S. Ridella, "Minimizing Multimodal Functions of Continuous Variables with the 'Simulated Annealing Algorithm", *ACM Transactions on Mathematical Software*, **13**, 262-280, (1987).

[54] W. L. Goffe, G. D. Ferrier and J. Rogers, "Global Optimization of Statistical Functions with Simulated Annealing"**,** *Journal of Econometrics*, **60**, 65-99, (1994).

[55] G. S. Dulikravich, T. J. Martin, B. H. Dennis and N. F. Foster, "Multidisciplinary Hybrid Constrained GA Optimization", in *EUROGEN'99 - Evolutionary Algorithms in Engineering and Computer Science: Recent Advances and Industrial Applications* (eds. K. Miettinen, M. M. Makela, P. Neittaanmaki and J. Periaux), John Wiley & Sons, Jyvaskyla, Finland, 233-259, May 30 - June 3, (1999).

[56] B. N. Pshenichny and Y. M. Danilin, *Numerical Methods in Extremal Problems*, MIR Publishers, Moscow (1969).

[57] J. L. Zhou and A. Tits, "User's Guide for FFSQP Version 3.7: A Fortran Code for Solving Optimization Programs, Possibly Minimax, with General Inequality Constraints and Linear Equality Constraints, Generating Feasible Iterates", *Institute for Systems Research*, Univ. of Maryland, Tech. Report SRC-TR-92-107r5, (1997).