VINCENȚIU DUMITRU
Academy of Economic Studies
The Department of Economic
Cybernetics

Function Optimization without Evaluating Derivatives: Methods and Techniques, Flowcharts, Computer Subroutines and Numerical Results(I)

Part one of this paper is a review, following well known authors in the field, of some of the methods and techniques for function optimization, without evaluating derivatives. An attempt has been made to present the algorithms in such a way to facilitate the programming tasks. In part two, flowcharts, computer subroutines and numerical results will be presented.

METHODS AND TECHNIQUES

The evaluation of the derivative-when it is possible- and the use of data supplied by it may be expensive, at least from the point of view of time, both for man and the machine. This is an argument and even if it were not, or even if it should be opposed to problems linked with accuracy, speed, convergence, flexibility and especially with the demonstrations on the convergence of the method we can accept the question as a given problem.

The methods we propose to discuss about, generally, regard the problem of unconstrained optimization of a function of n variables, and, generally, they have in common the finding of a local optimum -more precisely, its approximation, by successive iterations starting from an initially given point and some initially given directions.

We mention, in brief, that the problem of the unconstrained optimization of a function has its importance and that from the point of view of solving numerical problems, it is used, for example, in solving equations or equation systems and, from the point of view of mathematical programming, intense studies have lately been made on the techniques of reducing a constrained problem to an unconstrained one. In this direction, the reader is asked to consult [1], [2], [3], [4], where, as a matter of fact, expositions are made on the subject tackled here.

A. THE CASE OF UNCONSTRAINED OPTIMIZATION OF A FUNCTION OF n VARIABLES *)

The method of Hooke and Jeeves [5], [6], though often quoted, — may be also from methodological considerations- has a high degree of empiricism. The method uses two kinds of moves: "exploratory search moves", trying to determine the local behaviour of the function and "pattern moves", by which — formally, we should say, they try to explore the global "shape" of the function, by a somewhat movement on the diagonal of the exploratory moves.

An iteration k starts from an initial point x_0^k and ends by determining a new initial point x_0^{k+1} which may be different or not from the preceding initial point and requires the following stages:

(i) calculate $f(x_0^k) = : F$

(ii) From point x_0^k we explore the behaviour of the function in its vecinity by "exploratory moves" of the given length s_i, on each of the directions of the axes of coordinates e_i, $i = 1 \dots n$, e_i being considered here the unitary vectors in space E^n, as follows:

Initialize $i = 0$

(a) Do $i := i + 1$ and determine point $x_i^k = x_{i-1}^k + s_i e_i$

(b) Calculate $f(x_i^k)$. *If* $f(x_i^k) < F$, attribute to F the value $f(x_i^k)$, $F := f(x_i^k)$ and repeat from (a).

(*b*) · If $f(x_i^k) \geqslant F$ point x_i^k established at (a) is considered unsatisfactory to the search process and another one is determined, on the same direction, but in an opposite way:

$$x_i^k := x_{i-1}^k - s_i e_i$$

If $f(x_i^k) < F$, we attribute to F the value $f(x_i^k)$, $F := f(x_i^k)$ and the process restarts from (a) · If again $f(x_i^k) \geqslant F$ then, for the moment, we give up the searching of a better point on direction i, the components of x_{i-1}^k, $x_i^k := x_{i-1}^k$ are attributed to x_i^k and the process is repeated from (a)until the exhaustion of all n directions of search.

(iii) Point x_n^k is reached at the end of the exploring stage. After this stage is completed, a formal move is carried out, somewhat on the diagonal of the successful movements made within the exploring moves, by computing

$$x_0^f = x_n^k + (x_n^k - x_0^{k-1}) = 2x_n^k - x_0^{k-1}$$

a point in which stages (i) and (ii) are repeated and from where, point x_n^f is reached.

(iv) If $f(x_n^f) < f(x_n^k)$ then $x_0^{k+1} = x_n^f$, and if $f(x_n^k) \leqslant f(x_n^f)$, then $x_0^{k+1} := x_n^k$ and a new iteration begins.

If the exploratory moves no longer lead to points in which the value of the function is lower than in the initial point, then the exploring process

*) In the following, when no other specification is given, we suppose that the minimum is sought.

restarts with the diminished steps s_i and ends when they reached below the before established limit.

Part two of this article gives a logical flowchart and a numerical example.

Rosenbrock's method [7] is quoted as having obtained essentially better results than Hooke's and Jeeves's method and, though preceding it, methodologically speaking, it can be considered as its development. Whereas, we have no formal moves here, however, a new set of exploratory directions is determined at each iteration. These n directions are mutually orthogonal here too, but they no longer are the directions of the co-ordinate axes and they are chosen so as to reflect the progress scored at the previous iteration.

We shall describe such a current iteration k:

At iteration k we have an initial point x_0^k, a matrix D^k, $(n \times n)$, of directions, whose column vectors d_i^k, $i = 1 \ldots n$, define a set of n mutually orthogonal directions on which exploratory moves will be carried out, the total length of successful steps on each direction will be noted by λ_i^k; $\{s_i^k\}$ is the set of numbers defining the length of the initial steps to be taken on each direction d_i^k and $\alpha > 1$ and $0 < \beta < 1$ are two constant values with which, by multiplication, the length of the step will change in case of success and in case of failure, respectively.

Initially, matrix D is usually chosen as being the unit matrix. We initialize $i = 0$, $\lambda_i = 0$

The stages of an iteration are the following:

(i) Calculate $f(x_0^k)$, $F := f(x_0^k)$

(ii) Establish a new testing point, on direction $i := i + 1$

$$x_i^k := x_{i-1}^k + s_i^k d_i^k$$

(a) If $f(x_i^k) \leqslant F$, then

$$F := f(x_i^k)$$

$$\lambda_i^k := \lambda_i^k + s_i^k$$

$$s_i^k := \alpha s_i^k$$

(b) If $f(x_i^k) > F$, then

$$x_i^k := x_{i-1}^k$$

$$\lambda_i^k := \lambda_i^k$$

$$s_i^k := -\beta s_i^k$$

(iii) The step (ii) is repeated at each iteration until on each direction we obtained at least a success followed by a failure, point x_0^{k+1} being finally obtained.

(iv) Determine the new exploring directions d_i^{k+1} in point x_0^{k+1}

If we note by

$$L^k = \begin{bmatrix} \lambda_1^k \\ \lambda_2^k \; \lambda_2^k \\ \ldots\ldots\ldots \\ \ldots\ldots\ldots \\ \lambda_n^k \; \lambda_n^k \ldots \lambda_n^k \end{bmatrix}$$

where, as they were calculated, λ_i^k represents the sum of successful steps obtained at iteration k on direction i, by its very way of formation, the column vectors of this matrix represent a set of n independent linear vectors.

Matrix $Q^{k+1} = D^k \cdot L^k$, in which D^k represents the matrix of directions at iteration k will also have the column vectors q_i^k lineary independent and will be used to determine the new directions.

As $x_0^{k+1} = x_0^k + \sum_{i=1}^{n} \lambda_i^k d_i^k$ it results that q_i^k is just $x_0^{k+1} - x_0^k$, namely it represents the move direction from x_0^k to x_0^{k+1}, respectively the direction in which the biggest success was scored. Setting out from this, a process of Gram-Schmidt orthogonalization of the vectors q_i is applied after the well known formulas:

$$\delta_i^{k+1} = q_i^{k+1} - \sum_{j=1}^{i-1} (q_i^{k+1} d_j^{k+1})\, d_j^{k+1}, \quad i = 1,2,\ldots,n,$$

where $d_i^{k+1} = \dfrac{\delta_i^{k+1}}{\| \delta_i^{k+1} \|}$ and, in this way, matrix D^{k+1} is obtained.

The method stops when a certain "convergence criterion" is satisfied, for example, when the length of the step has decreased below a certain limit or when the variation of function from one iteration to another has decreased below a certain limit.

As regards parameters α and β, Davies in [8] mentions as typical values $\alpha = 3$ and $\beta = 0.5$.

Rosenbrock's method has generated natural developments among which we shall mention the method of Davies, Swann and Campey for whom I have disposed of brief references in [8] and [9] and Powell's method [10].

In *the method of Davies, Swann and Campey (D.S.C.)*, the directions generated at iteration k, according to which the optimization of the function at iteration $k + 1$ is made, are the same as in Rosenbrock's method, namely:

We consider: $q_i^{k+1} = x_n^k - x_{i-1}^k, \; i = 1,\ldots,n,$

which can be easily seen that it is the same as in stage (iv) of Rosenbrock's method and afterwards these directions are orthogonalized. What is different is that here, points x_i^k are no longer found by exploratory moves, but by linear minimizations along each direction d_i^k, that is

$$x_i^k = x_{i-1}^k + \lambda_i^k d_i^k, \; i = 1,\ldots,n,$$

where λ_i^k is chosen so that $f(x_i^k)$ is the minimum of $f(x)$ along direction d_i^k passing through point x_{i-1}^k.

With the aid of the thus established steps λ_i^k we can also write an alternative formula for q_i^{k+1} and, namely

$$q_i^{k+1} = \sum_{j=i}^{n} \lambda_j^k d_j^k, \quad i = 1,\ldots n,$$

which shows once more the identity at this point with Rosenbrock's method. Just as in it, the initial directions may be taken as the directions of the coordinate axes.

If those expounded so far suggest a somewhat facile technique of developing new algorithms, we cannot say the same about the problem of the convergence of algorithms. The one raising it almost vehemently, when a new method "yielding good results" is presented to him, is P. Wolfe in [11] and [12] to whom we send the reader. We confine ourselves to introducing here *the notion of quadratic convergence.*

We say that an algorithm is quadratically convergent if applied to a quadratic function $f(x) = \frac{1}{2} x\,Ax + bx + c$, with A symmetric and positiv definite matrix, to ensure the existence of the minimum- it leads to solutions in a finite number of iterations, generally n.

Powell's algorithm [10], described below, is a quadratically convergent algorithm.

(1) Just as in the D.S.C. algorithm at each iteration k one starts from an initial point x_0^k and reaches point x_n^k, with the aid of n succesive linear minimizations along directions d_i^k namely successively passing through points

$$x_i^k = x_{i-1}^k + \lambda_i^k d_i^k,\ i = 1, \ldots, n,$$

where λ_i^k minimizes $f(x_i^k)$ along direction d_i^k passing through point x_{i-1}^k (x_0^0 is chosen as being the best approximation of the minimum and, d_i^0 the directions of the co-ordinate axes).

(2) At step 2 of iteration k, we determine λ^k so that $f(x_n^k + \lambda^k\{x_n^k - x_0^k\}) =$ minimum and with thus calculated λ^k we determine $x_0^{k+1} = x_n^k + \lambda^k\{x_n^k - x_0^k\}$.

(3) At step 3, we determine the new set of directions d_i^{k+1} through the formulas:

$$d_i^{k+1} := d_i^k \quad i = 1, 2, \ldots, (n-1),$$

$$d_n^{k+1} := x_n^k - x_0^k$$

Let us show, following the demonstration in [10] that the algorithm is quadratically convergent. The demonstration results with the aid of two theorems:

Theorem 1. If $q_1, q_2 \ldots q_m$, $m \leqslant n$, are m directions mutually conjugate with respect to matrix A, symmetric and positive definite — a necessary condition for the existence of the minimum and the calculus reasoning and not for argueing the demonstration — then the minimum of the quadratic form

$$f(x) = \frac{1}{2} x\,Ax + b\,x + c,$$

where x is a point in the m dimensional space, containing x_0 and the directions $q_1, q_2, \ldots, q_m$, is found in the point

$$x^* = x_0 + \sum_{i=1}^{m} \lambda_i q_i,$$

where λ_i are so as to minimize $f(x_0 + \lambda_i q_i)$; thus, the minimum may be found through m minimization, along each direction only once, irrespective of order.

Suppose that x^* is a minimum point and it is obtained by successive moves on directions q_i of the length λ_i
We have

$$f(x^*) = f\left(x_0 + \sum_{i=1}^{m} \lambda_i q_i\right) = \frac{1}{2}\left\{x_0 + \sum_{i=1}^{m} \lambda_i q_i\right\} A \left\{x_0 + \sum_{i=1}^{m} \lambda_i q_i\right\} +$$

$$+ b\left\{x_0 + \sum_{i=1}^{m} \lambda_i q_i\right\} + c = \sum_{i=1}^{m}\left\{\frac{1}{2}\lambda_i^2 q_i A q_i + \lambda_i q_i (A x_0 + b)\right\} + f(x_0)$$

the terms in $\lambda_i \lambda_j$, $i \neq j$ disappearing because of conjugacy condition $q_i A q_j = 0$

Obviously, at x^*, $\frac{\partial f(x^*)}{\partial \lambda_i} = 0$, $i=1, ..., m$, which implies $\frac{\partial}{\partial \lambda_i}\left\{\frac{1}{2}\lambda_i^2 q_i A q_i + \right.$

$\left. + \lambda_i q_i (A x_0 + b)\right\} = 0$, $i = 1, ..., m$, and, thus $\lambda_i = -\frac{q_i (A x_0 + b)}{q_i A q_i}$, from where it can be seen that the value of λ_i is independent from the other terms of the function and that it can be found by seeking $min\, f(x_0 + \lambda_i q_i)$

Indeed,

$$f(x_0 + \lambda_i q_i) = \frac{1}{2}(x_0 + \lambda_i q_i) A (x_0 + \lambda_i q_i) + b(x_0 + \lambda_i q_i) + c =$$

$$= \frac{1}{2}(x_0 + \lambda_i q_i)(A x_0 + \lambda_i A q_i) + b x_0 + \lambda_i b q_i + c =$$

$$= \frac{1}{2}(x_0 A x_0 + \lambda_i x_0 A q_i + \lambda_i q_i A x_0 + \lambda_i^2 q_i A q_i) + \lambda_i b q_i + b x_0 + c =$$

$$= \frac{1}{2}\lambda_i^2 q_i A q_i + \lambda_i q_i (A x_0 + b) + \frac{1}{2} x_0 A x_0 + b x_0 + c =$$

$$= \frac{1}{2}\lambda_i^2 q_i A q_i + \lambda_i q_i (A x_0 + b) + f(x_0)$$

from where, derivating in respect with λ_i; and solving the equation obtained, the same values as above result for λ_i.

Theorem 2.

If x_1 is the minimum of $f(x) = \frac{1}{2} x A x + bx + c$ in the space containing direction q, on the direction q, and x_2 is the minimum of $f(x)$ on the same direction, but starting from a different point, then directions q and $(x_1 - x_2)$ are conjugated with respect to A, that is $qA(x_1 - x_2) = 0$.

Indeed, direction q and the gradient of the function $\nabla f(x) = Ax + b$ in the minimum points are 2 orthogonal vectors. It results:

$$q(A x_1 + b) = 0$$

$$q(A x_2 + b) = 0$$

From where:

$$qA(x_1 - x_2) = 0$$

The convergence to the minimum of the quadratic function $f(x) = \frac{1}{2} x A x + bx + c$ in the Powell's algorithm can now be shown by induction.

Suppose that after k iterations, the last k directions generated in the course of iterations, d_{n-k+1}^{k+1}, d_{n-k+2}^{k+1}, ..., d_n^{k+1}, after which optimization is made at iteration $k+1$ are conjugated.

As they are the last k mutually conjugate directions of minimization, it results that at iteration $k+1$, when the $k+1$ conjugate direction is obtained, point x_n^{k+1} being a minimum after the last directions, it will be a minimum after the last conjugate direction produced at iteration k. On the other hand, the initial point at iteration $k+1$, x_0^{k+1} was defined so as it achieved the minimum after the direction generated at iteration k. According to theorem 2, it results that direction $x_n^{k+1} - x_0^{k+1}$ generated at iteration $k+1$ is conjugate with the direction generated at iteration k. Completing the induction by a reasoning similar for the direction generated at the second iteration it results that all generated directions are mutually conjugated.

We should now point out that at the end of iteration $n+1$ the optimum of $f(x) = \frac{1}{2} x A x + bx + c$ is effectively attained. As results from the condition of annuling the gradient, it is found in point $x^* = -A^{-1} b$.

To avoid confusions of notation, we shall note by p_i the n conjugate directions obtained after iteration n and by x_0 the initial point at the beginning of iteration $n+1$.

Optimizing after each direction at the end of iteration $n+1$, the point x_n which is obtained, is according to theorem 1 the minimum point.

The result may be obtained also directly:

$$x_n = x_0 + \sum_{i=1}^{n} \lambda_i p_i$$

where, taking into consideration the property of conjugacy.

$$\lambda_i = - \frac{p_i(Ax_0 + b)}{p_i A p_i}$$

it results:

$$x_n = x_0 - \sum_{i=1}^{n} \frac{p_i(Ax_0 + b)}{p_i A p_i} \cdot p_i = x_0 -$$

$$- \sum_{i=1}^{n} \frac{p_i A x_0}{p_i A p_i} p_i - \sum_{i=1}^{n} \frac{p_i A(A^{-1}b)}{p_i A p_i} \cdot p_i = x_0 - x_0 - A^{-1}b = -A^{-1}b$$

namely the minimum of the quadratic form $f(x)$ and this, if we take into consideration the fact that if p_i are mutually conjugated, for any vector v we have the identity $v = \sum_{i=1}^{n} \frac{p_i A v}{p_i A p_i} p_i$.

The quadratically convergence property of the algorithms in which we are optimizing on conjugate directions being very important, explains why attempts were made and continue to be made in elaborating algorithms, in which the optimization directions are mutually conjugate.

Fletcher in [9] points out that Smith affirms in [13] that when Rosenbrock's algorithm is applied for minimizing a quadratic function the generated directions d_i^k tend, with the growth of k, to become parallel with the vectors of matrix A, vectors which represent a particular case of conjugate directions and, thus, the method would have a certain quadratic convergence property, which would explain its succes in application.

If, generally, the methods described so far can be characterized as sequential methods of minimization after certain directions — n, in general — *the simplex method for unconstrained minimization* described for the first time by *Himsworth, Spendley and Hext* in [14] and developed by *Nelder and Mead* in [15], uses the figure generated by $n + 1$ points in the dimensional n space-a figure called simplex. In this figure, estimating the value of the function at the vertices of the simplex and replacing the vertex in which the function has the highest value (we suppose again that the minimum is sought) by a point in which the function has a lower value, they try to shift the initially generated simplex to the region of the minimum. Once getting here, this is contracted so that the values of the function at the vertices of the simplex should allow the approximation of the minimum and its possible determination by interpolating a quadratic function which is supposed to approximate adequatelly the function in the vecinity of the minimum.

Nelder's and Mead's version which we present here allows the simplex to vary in form and size keeping it in accordance with the local geometry of the function. The method is given in [15], but we will describe following [2] and [8]:

Three operations: reflection, expansion and contraction are defined in the process of changing the simplex, simplex defined by the points x_1, $x_2, \ldots, x_{n+1}$.

The following notations are made:

x_h: the vertex corresponding to $f(x_h) = \max\limits_i f(x_i), i = 1, \ldots, n + 1$

x_s: the vertex corcesponding to $f(x_s) = \max\limits_i f(x_i), i \neq h,\ i = 1, \ldots, n + 1$

x_l: the vertex corresponding to $f(x_e) = \min\limits_i f(x_l),\ i = 1, \ldots, n + 1$

x_0: the centroid of all points x_i, $i \neq j$, given by the formula

$$x_0 = \frac{\sum\limits_{i=1}^{n+1} x_i - x_h}{n}$$

The above operations are defined by:

(a) Reflection represents the image of x_h towards the centroid, given by $x_r = x_0 + \alpha(x_0 - x_h)$ where α is a given positive reflection factor.

(b) Expansion is the operation by which, in case that the value of the function in x_r is better than in x_h, one tries to find an even better value by replacing point x_r by point x_e, in the idea that by extending the move on direction $x_r - x_0$, which is the same as direction $x_0 - x_h$, we might obtain a point in which the function has an even lower value:

$$x_e = x_0 + \gamma(x_r - x_0),\ \gamma \geqslant 1$$

c) The contraction operation by which the simplex is diminished in size is achieved either partially, on direction $x_h - x_0$, by replacing x_h by x_c, given by

$$x_c = x_0 + \beta(x_h - x_0) \quad 0 < \beta < 1$$

or totally, as it will be shown in the stages of the algorithm, by replacing all the other points of the simplex besides x_l by points half the distances from the respective vertices and x_l:

$$x_i := \frac{x_l + x_i}{2} \quad i = 1, \ldots, n+1$$

An iteration k starts with the calculus of the values of function in the $n+1$ vertices x_i^k, $i = 1, \ldots, n+1$, of simplex k and ends by generating the simplex $k+1$. We calculate x_r^k and x_0^k with the aid of the given formulas. With the notations made, in the course of an iteration, the following mutually disjunct situations may occur generated by the value of $f(x_r^k)$, in which the iteration ends as follows:

(i) $$f(x_l^k) \leqslant f(x_r^k) \leqslant f(x_s^k)$$

In this case:

$$x_i^{k+1} := x_i^k \quad i \neq h,\ i = 1, .., n+1$$

$$x_h^{k+1} := x_r^k$$

(ii) $$f(x_r^k) < f(x_l^k)$$

In this case, we may expect to find in the direction on which the reflection was made a better value, for calculating:

$$x_e^k = x_0^k + \gamma(x_r^k - x_0^k),\quad \gamma \geqslant 1$$

a) If $f(x_r^k) \leqslant f(x_e^k)$

$$x_i^{k+1} := x_i^k \quad i \neq h,\quad i = 1 \ldots n+1$$

$$x_h^{k+1} := x_r^k$$

b) If $f(x_e^k) < f(x_r^k)$

$$x_i^{i+1} := x_i^k \quad i \neq h,\quad i = 1, \ldots, n+1$$

$$x_h^{k+1} := x_e^k$$

(iii) $f(x_r^k) > f(x_s^k)$. We have no reason to be very satisfied with the result of the reflection towards the centroid.

It is only natural to believe that we shall be able to obtain a better value either between the centroid and the reflected point, in other words to consider that the reflection coefficient was too big, or between x_h^k and the centroid. Such a point will be obtained by the contraction operation, just as:

a) $f(x_r^k) < f(x_h^k)$

We calculate:

$$x_c^k = x_0^k + \beta(x_r^k - x_0^k), \quad 0 < \beta < 1$$

If $f(x_c^k) \leqslant f(x_s^k)$, the contraction was successful

$$x_i^{k+1} := x_i^k \quad i \neq h, \quad i = 1 \ldots n+1$$

$$x_h^{k+1} := x_c^k$$

If $f(x_c^k) > f(x_s^k)$ point x_c^k as well as x_r^k is still the point with the highest value; we remain disatisfied with the operations made on the centroid and then, we carry out the total contraction operation around x_l^k at step c), where iteration ends.

b) $f(x_r^k) \geqslant f(x_h^k)$

A contraction is made toward x_h^k and we shall have:

$$x_c^k = x_0^k + \beta(x_h^k - x_0^k), \; 0 < \beta < 1,$$

If $f(x_c^k) \leqslant f(x_s^k)$, the contraction was successful and the iteration ends by

$$x_i^{k+1} := x_i^k \; i \neq h, \quad i = 1, \ldots, n+1$$

$$x_h^{k+1} := x_c^k;$$

If $f(x_c^h) > f(x_s^k)$, the contraction was unsuccessful and the operation of total contraction is effectuated at c).

c) The operation of total contraction is made around point x_l^k, in which the function has the lowest value and through it the new simplex is made up of points:

$$x_i^{k+1} := \frac{x_l^k + x_i^k}{2} \quad i = 1,2, \ldots, n+1.$$

As regards the value of parameters α, β and γ, Nelder and Mead suggest $\alpha = \gamma = 1$, $\beta = \frac{1}{2}$. By extending, as it will be seen, this algorithm in the case of constrained optimization, Box [16] considers $\alpha = 1.3$.

As a convergence test, Nelder and Mead use

$$S_k = \left\{ \sum_{i=1}^{n+1} [f(x_i^k) - \bar{f}_k]/n \right\}^{\frac{1}{2}} < \varepsilon$$

where $\bar{f}_k = \dfrac{\sum_{i=1}^{n+1} f(x_i^k)}{n+1}$, that is the standard deviation of the values of the function at the n + 1 vertices of the simplex. In [8], Davies speaks about

a much severer criterion which might be applied, namely to calculate S after the carrying out of a number of, before established, m iterations, $k_p = p \cdot m$ and if:

$$|S_{k_p} - S_{k_{p+1}}| < \varepsilon_1$$

and

$$|\bar{f}_{k_p} - \bar{f}_{k_{p+1}}| < \varepsilon_2$$

where ε_1 and ε_2 are two arbitrary small numbers, established beforehand, then the method should be considered as convergent.

B. THE CONSTRAINED OPTIMIZATION OF A FUNCTION OF n VARIABLES

The problem of the constrained optimization of a function of n variables, although representing the main problem of mathematical programming does not form the object of this paper. However, to show the importance of unconstrained optimization, we shall enumerate some methods of solving a constrained problem by bringing it to an unconstrained problem and we shall also give the extension in the constrained case of one of the described algorithms.

A detailed description of the methods for constrained optimization of problems through the sequential solving of a number of unconstrained problems is given in [1] and [2] · In [3] and [4] one can found new valuable contributions for solving the problem.

Obviously, in a problem in which some of the constraints are given by equalities they can be used for eliminating some of the variables on which no sign constraints are made.

The method of variable transformations can be sometimes successfuly used for eliminating constraints and Box, in [17] gives many suggestions which he recommends even if the model appears more complicated in the new variables. The problem is also tackled in [18].

It is obvious that [8], for example, a constraint of the form

$$x_1 \geqslant a$$

can be eliminated with the aid of one of the transformations

$$x_1 = a + y_1^2 \quad \text{or} \quad x_1 = a + e^{y_1}$$

and a constraint of the form $a \leqslant x_1 \leqslant b$ by transforming

$$x_1 = a + (b - a)\ sin^2 y_1.$$

The method of Lagrange's multipliers for constrained problems given by equalities as well as calculus procedure suggested by Arrow and Uzawa in [19], as a result of the development of the theory of Lagrange's multipliers by Kuhn-Tucker in the case of constraints given by inequalities is also leading to an unconstrained problem. For references, one can consult [1], [2], [20], or in Romanian literature [21]. As regards this method and the method of *penalizing functions or that of Carroll's response surface*, we mention as references [1], [2], [3] and [4].

As I said at the beginning of this paragraph, besides these techniques of transforming a constrained problem into an unconstrained one, some of the algorithms described in paragraph A were completed in view of their use for constrained problems. Some of these developments use derivatives of the first even the second order and thus, they will not be described here:

Rosenbrock's algorithm applied to a constrained problem given in the form:

$$l_i \leqslant x_i \leqslant u_i \quad i = 1, ..., n,$$

$$l_i \leqslant x_i \leqslant u_i \quad i = n+1, ..., r,$$

where x_i, $i = n+1, ..., r$, are functions of x_i, is applied in the form described for the unconstrained problem until one of the variables get in the neighbourhood of the boundary region whose width is considered to be $10^{-4}(u_i - l_i)$.

In this situation, in order not to alow the point to get out the domain of solutions, supposing that the minimum would really be outside it, the value of the function in point x_i^k, $f(x_i^k)$ is replaced by

$$f_1(x_i^k) = f(x_i^k) - [f(x_i^k) - f(x_i^p)] \cdot [3\alpha_k - 4\alpha_k^2 + 2\alpha_k^3]$$

where $\alpha_k = \dfrac{l_i + 10^{-4}(u_i - l_i) - x_i^k}{10^{-4}(u_i - l_i)}$ and x_i^p is the last value of x_i on direction i and which is found outside the vecinity of restriction i.

Because, when α increases from zero to one, it means that variable x_i^k is moving from inwards to outwards of the domain and, we suppose that in this direction the function is continuously decreasing: $f(x_i^k) \leqslant f(x_i^p)$; just as for $\alpha = 0$, $f_1(x_i^k) = f(x_i^k)$ and for $\alpha = 1$, $f_1(x_i^k) = f(x_i^p)$, it results that in this vecinity we no longer optimize the continuously decreasing function $f(x)$, but the function $f_1(x)$, which has a local minimum in this vecinity.

If, at a given moment x_i^k, however, gets outside the domain, a number of succesive moves is carried out, backwards and forwards on that direction, by diminishing the step until x_i^k is brought to the inward vecinity of the constraint.

Davies and Swann [16] affirm that the above described technique proved "reliable"; Swann [22] points out that if the search algorithm is considered to be other than Rosenbrock's algorithm, this technique does not yield very good results.

The D.S.C. method with linear constraints [16], described for the first time in [22], extends the D.S.C. method for unconstrained optimization so that at each iteration k, the directions d_i^k should be chosen in such a way that the points x_i^k obtained should satisfy the linear constraints:

$$A^T x = b$$

where A is a matrix $(n \times p)$, with columns $c_1, c_2, ..., c_p$.

For this, at each iteration k, the first d_i^{k+1} directions are considered equal to c_i, $d_i^{k+1} := c_i$, $i = 1, ..., p$. The other $n - p$ directions are calculated according to the formula:

$$d_i^{k+1} = P_{i-1} q_i^{k+1} \quad i = p+1, ..., n$$

where $q_i^{k+1} = x_n^k - x_{i-1}^k$, and $P_i = I - A_i(A_i^T A_i)^{-1} A_i^T$, where A_i is the submatrix of A, with $(n \times i)$.

Indeed, if $d_1^{k+1}, d_2^{k+1}, \ldots, d_p^{k+1}$ have already been chosen, then for $i = p + 1, \ldots, n$,

$$d_i^{k+1} = q_i^{k+1} - A_{i-1} l_i$$

and l_i must be so that d_i^{k+1} should be orthogonal to all the previous directions, that is to each column of A_{i-1}, that is we must have

$$A_{i-1}^T d_i^{k+1} = 0$$

hence $A_{i-1}^T q_i^{k+1} - A_{i-1}^T A_{i-1} l_i = 0$

and, thus $l_i = (A_{i-1}^T A_{i-1})^{-1} A_{i-1}^T q_i^{k+1}$

It results

$$d_i^{k+1} = q_i^{k+1} - A_{i-1}(A_{i-1}^T A_{i-1})^{-1} A_{i-1}^T q_i^{k+1}$$

$$d_i^{k+1} = [I - A_{i-1}(A_{i-1}^T A_{i-1})^{-1} A_{i-1}^T] q_i^{k+1}$$

That is, $d_i^{k+1} = P_{i-1} q_i^{k+1}$

If we take into consideration that only $n - p$ directions are used at each iteration, it results that only $n - p$ vectors q_i will be independent; it is thus possible that $P_{i-1} q_i = 0$, in which case this direction is not taken into consideration.

The Complex method is an extension, in case of constrained optimization, of the simplex method and was elaborated by J.M. Box [23], for taking into consideration constraints in the form of inequalities of the form:

$$g_i \leqslant x_i \leqslant h_i \qquad i = 1, 2, \ldots, n$$

$$G_j \leqslant \varphi_j(x) \leqslant H_j \qquad j = 1, 2, \ldots, m,$$

where g_i, h_i, G_j, H_j are constant or functions of $x = [x_1 \ldots x_n]$.

One of the diferences between the complex and simplex methods is that Box extends the simplex to a complex of $k \geqslant n + 1$ points. Analysing "the convergence" for two problems, Box affirms [23] that he has unsuccessfully tried to apply Rosenbrock's constraid method.

He stands for $k = 2n$ and $\alpha = 1.3$. For $\alpha = 1.3$, a super-reflection is produced and by this, the expansion operation is eliminated.

Suppose that an initial point which satisfies all constraints is known. The other points of the initial complex are generated, in turn, with the aid of formula

$$x_i^k = g_i + r_i^k (h_i - g_i)$$

where r_i^k are pseudo-random numbers, rectangularly distributed on the interval [0.1]. (Here x_i^k means the component i of point x^k).

Obviously, point x^k of components x_i^k satisfies the explicit constraints. If it does not satisfy the implicit constraints, it is contracted with $\beta = \frac{1}{2}$, half the distance between it and the centroid of the points which already satisfy all constraints. The contraction operation is repeated until the obtained point satisfies all constraints, thus generating all the vertices of the complex.

Box describes his algorithm as follows:
The function is evaluated at each vertex and point x_h with the highest value is super-reflected with the aid of formula

$$x_r = x_0 + \alpha(x_0 - x_h) \quad \alpha \geqslant 1$$

where x_0 is the centroid of all other points except x_h. If this point has also the highest value $f(x_r) > f(x_s)$ it is replaced by another point half the distance between it and the centroid of the remaining points and the procedure is repeated until a point with a value $f(x_r) \leqslant f(x_s)$ is obtained, or, some constraints are violated.

If such a vertex does not satisfy explicit constraint then the variables that violate this constraint, are reset to appropriate boundary values, equals to $\pm 10^{-5}$ inside the corresponding limits given to then by the constraints.

If an implicit constraint is violated, the vertex concerned is moved half the distance between it and the centroid of the other vertices.

The stop criterion suggested by Box is that of stopping the iterative method when five consecutive values were obtained of the function which differs with an accuracy equal to that of the length of "the word" used by the computer.

Besides the performances related to the more or less efficient solving of some very nonlinear problems, due to the generation of a number of $k > n + 1$ points it seems that Box's complex method has the advantage of finding, with greater probability, the global optimum.

C. METHODS OF OPTIMIZING FUNCTIONS OF A SINGLE VARIABLE WITHOUT USING THE DERIVATIVE

Methodologically, this paragraph should be found at the beginning of the paper. It would have been more difficult, we believe, to convince the reader of the fact that it is not only a variation on a given theme, but, a real problem occuring, for example, in the algorithms of Powell or Davies, Swann and Campey for optimizing a function of n variables without evaluating the derivative. Indeed, in these algorithms it is necessary to optimize the function according to some directions:

$$min\,(max)\ f(x_{i-1}^k + \lambda_i^k d_i^k)$$

where x_{i-1}^k, d_i^k are known.

Thus, consider $f(x)$ a function of a single variable for which it is necessary to find, for example, the minimum. One of the most frequently used methods is that of approximating the minimum by a number of points which, in their turn, are the minima of the polynominals of the second order with which, by interpolation, the function is approximated; the method is in concordance with the hypothesis that in the neighbourhood of the minimum the function may be approximated sufficiently well by a polynominal of the second order, respectively, by a quadratic form for the functions of several variables.

I. *The minimizing method with the aid of interpolation polynominals* marks several developments, among which we mention two:

a) Davies, Swann and Campey [22] present the following technique based on the interpolation of the function in three points situated in successive intervals in which the minimum is localized:

Considering a step s^k and a point x_0^k:

(i) Evaluate the function in x_0^k and in $x_0^k + s^k$. If $f(x_0^k + s^k) \leqslant f(x_0^k)$, then the function is evaluated in points $x_i^k = x_0^k + 2^{i-1} \cdot s^k$ until $f(x_i^k) > f(x_{i-1}^k)$; if $f(x_0^k + s^k) > f(x_0^k)$, then the function is evaluated in point $f(x_0^k - s^k)$ and, supposing that $f(x_0^k - s^k) \leqslant f(x_0^k)$. we determine points $x_i^k = x_0^k - 2^{i-1} \cdot s^k$ until $f(x_i^k) > f(x_{i-1}^k)$. If neither $f(x_0^k - s^k)$ is smaller than $f(x_0^k)$, then the step is diminished, for example, $s^k := \frac{s^k}{2}$ and, the process is repeated from(i).

(ii) Points x_i^k and k_{i-1}^k determined at step (i) determine an interval containing the minimum point. To be able to approximate this minimum through the minimum of a quadratic function a third point is necessary, which is taken in the middle of interval $[x_i^k, x_{i-1}^k]$.

To facilitate the notation, we shall re-attribute the indices so that $x_1 := x_{i-1}^k$, $x_3 := x_i^k$, The third point will be $x_2 = \frac{x_1 + x_3}{2}$.

(iii) The minimum point is approximated in

$$x_0^{k+1} = x_2 - \frac{x_3 - x_1}{4} \cdot \frac{f(x_3) - f(x_1)}{f(x_3) - 2f(x_2) + f(x_1)}$$

and the method is repeated from (i)until, for example, $s^k < \varepsilon$, or until $2^i s^k$ is sufficiently big to infer that the function admitts no minimum.

It is pointed out in [2] that the algorithm is convergent and that the convergence is approximately of the order 1.3.

b) Powell [10] is substantiating his algorithm also on the idea of minimizing a quadratic form determined by the value of the function in three points.

Noting by x_0^k a point in E^n, by d^k a direction in E^n, Powell describes the following optimization method along the straight line $x_0^k + \lambda d^k$:

(i) Evaluate $f(x_0^k)$ and $f(x_0^k + sd^k)$, where s is a given step, smaller than a given "m" value as the upper bound of the step at x_0^k at the searched minimum point.

If $f(x_0^k + sd^k) < f(x_0^k)$, then calculate the value of the function in point $x_0^k + 2sd^k$ and the points x_0^k, $x_0^k + sd^k$, $x_0^k + 2sd^k$. are kept for interpolation.

If $f(x_0^k + sd^k) > f(x_0^k)$, calculate the value of the function in point $x_0^k - sd^k$ and the points $x_0^k + sd^k$, x_0^k, $x_0^k - sd^k$ are kept for interpolation.

(*ii*) With the aid of the 3 points obtained at point (*i*) the minimum of $f(x)$ is approximated as being the minimum of the interpolation polynominal of the second order passing through these points.

The extreme point of a polynominal of the second order in t, which passes through points $[t_1, f_{t_1}(x_0^k + t_1 d^k)], [t_2, f_{t_2}(x_0^k + t_2 d^k)]$ and $[t_3, f_{t_3}(x_0^k + t_3 d^k)]$ is given by $t = \frac{1}{2} \cdot \frac{(t_2^2 - t_3^2) f_{t_1} + (t_3^2 - t_1^2) f_{t_2} + (t_1^2 - t_2^2) f_{t_3}}{(t_2 - t_3) f_{t_1} + (t_3 - t_1) f_{t_2} + (t_1 - t_2) f_{t_3}}$

and it is a minimum if

$$\frac{(t_2 - t_3)f_{t_1} + (t_3 - t_1)f_{t_2} + (t_1 - t_2)f_{t_3}}{(t_1 - t_2)(t_2 - t_3)(t_3 - t_1)} < 0.$$

Replacing in the above formula $t_1 = 0$, $t_2 = s$, $t_3 = 2s$, or $t_1 = s$, $t_2 = 0$, $t_3 = -s$ and the corresponding values for $f_{t_i} = f(x_0^k + t_i d^k)$, we determine t and thus, a new point $x_0^k + td^k$.

If the extreme point is a maximum, or if $t > m$, m being the maximum allowed value of the step, the method is repeated with the points left after the remotest point from $x_0^k + td^k$ was eliminated.

If $t < m$, the minimum condition is satisfied and if $\min_i | t_i - t | < \varepsilon$, a given value for accuracy, then considering λ equal to that t_i which achieves the minimum, one considers the procedure concluded and the minimum is considered to be in point $x_0^k + \lambda d^k$.

If $\min_i | t_i - t | \geqslant \varepsilon$, the procedure is repeated with the points left of the four now established points: $x_0^k + t_1 d^k$, $x_0^k + t_2 d^k$, $x_0^k + t_3 d^k$, $x_0^k + td^k$, after we have eliminated the point in which the function has the highest value; but, this condition is not automatically applied being subordinated to the selection of three points so that they should determine an interval which should contain the minimum, that is, they should satisfy a relation of the form:

$$f(x_0^k + t_j d^k) > f(x_0^k + t_q d^k) < f(x_0^k + t_r d^k).$$

In [24] is given an optimization subroutine of a function after a given direction, written in *fortran* by Melanie Lenard who use Powell interpolating method, which can be successfully used in the programs for solving nonlinear problems using algorithms which require optimizations after given directions.

II. *If the function of a single variable is unimodal* on the interval $[a, b]$ on which the minimum is searched, then it can be shown that the most efficient algorithm, to the effect of the evaluation of an ever smaller number of the function, is an algorithm in which the interval in which the minimum is situated is determined *with the aid of Fibonacci's numbers.* This algorithm together with the algorithm of the golden section, are described in [2] after which we make the presentation.

Both these algorithms are based on the observation that if f(x) is unimodal in the interval $[a, b]$, then for localizing the minimum or maximum of the function in a subinterval of $[a, b]$ it is necessary to evaluate the function in two points inside the interval $[a, b]$, namely:

if, for example, if we search the minimum and $a < x_1 < x_2 < b$, then
if $f(x_1) > f(x_2)$, the minimum is found in the interval (x, b];
if $f(x_1) < f(x_2)$, the minimum is found in the interval $[a, x_2)$.

In both algorithms, considering the interval $[a^k, b^k]$ at an iteration k points x_1^k and x_2^k are determined so that

$$x_2^k - a^k = b^k - x_1^k = r_k(b^k - a^k) \qquad (1)$$

The difference consists in the fact that in the algorithm of the golden section, $r_k = \frac{\sqrt{5}-1}{2}$ is constant for all iterations, while in that based on Fibonacci's numbers, r_k is variable with each iteration, namely $r_k = \frac{F_{N-k}}{F_{N-k+1}}$, where F are Fibonacci's numbers whose form will be determined immediately.

It is clear that the best algorithm will be that in which, after the evaluation of the function of N times the length of the initial interval $[a^1, b^1]$ will be reduced to $\delta_N = \frac{b^1 - a^1}{F_N}$ where F_N is the length of the longest interval which as a result of evaluations of the function is reduced to the unit interval.

We shall note by $[a^1, b^1]$ of length F_N this biggest interval which may be reduced in N evaluations of functions at the unit interval and consider points x_1 and x_2, $x_1 < x_2$. If the minimum is found in the subienterval $[a^1, x_1)$, we have at our disposal, at most N-2 evaluations of functions for reducing this interval to the unit interval; so that,

$$x_1 - a^1 \leqslant F_{N-2}$$

If the minimum is found in the interval $(x_1, b^1]$, we have at our disposal N-1 evaluations of function for reducing this interval because x_2 is found in this interval and we know the value of the function in x_2; thus,

$$b^1 - x_1 \leqslant F_{N-1}$$

Adding these two inequalities, member by member, we have

$$b^1 - a^1 \leqslant F_{N-1} + F_{N-2}$$

that is

$$F_N \leqslant F_{N-1} + F_{N-2}$$

and, obviously F_i is maximum when it attains its highest bound

$$F_N = F_{N-1} + F_{N-2}$$

which shows that F_i are Fibonacci's numbers.

As by zero evaluations of function or by a single evaluation the reduction of any interval is impossible, it results that $F_0 = F_1 = 1$, the length of the unit interval.

After we have seen that F_i are Fibonacci's numbers, the algorithm is defined at each iteration k by relations:

$$x_1^k = \frac{F_{N-k-1}}{F_{N-k+1}}(b^k - a^k) + a^k, \quad x_2^k = \frac{F_{N-k}}{F_{N-k+1}}(b^k - a^k) + a^k$$

We should point out that after N evaluations, the length of the interval is

$$\delta_N = \frac{b^1 - a^1}{F_N}$$

Indeed, from the first relation of determining x_1^k given by the algorithm results

$$a^k - x_1^k = -\frac{F_{N-i-1}}{F_{N-i+1}}(b^k - a^k)$$

Hence, by successive transformations, we have:

$$b^k - b^k + a^k - x_1^k = -\frac{F_{N-i-1}}{F_{N-i+1}}(b^k - a^k)$$

$$b^k - x_1^k = \left(1 - \frac{F_{N-i-1}}{F_{N-i+1}}\right)(b^k - a^k)$$

And, combining with the second relation,

$$b^k - x_1^k = \frac{F_{N-i}}{F_{N-i+1}}(b^k - a^k) = x_2^k - a^k$$

Hence, replacing by recurrence, it results

$$b^N - x_1^N = x_2^N - a^N = \delta_N = \frac{b^1 - a^1}{F_N}$$

Which shows that the algorithm is optimal from the point of view of the number of evaluations of the function. It is obvious that if, for example, $f(x_1^k) < f(x_2^k)$, then $x_1^k = x_2^{k+1}$ a fact which quite easily results if we take into consideration the recurrence relations of Fibonacci's numbers and the fact that in this situation $b^{k+1} = x_2^k$.

As regards the values of F_i they are inferred from the solving of the recurrent equation:

$$F_N = F_{N-1} + F_{N-2}$$

with the conditions:

$$F_0 = F_1 = 1$$

for which a solution of the form is sought:

$$F_i = \alpha r_1^i + \beta r_2^i$$

By replacing under the initial conditions and taking into consideration the recurrence relations, we obtain the system

$$\begin{aligned} \alpha + \beta &= 1 \\ \alpha r_1 + \beta r_2 &= 1 \\ \alpha r_1^2 + \beta r_2^2 &= 2 \\ \alpha r_1^3 + \beta r_2^3 &= 3 \end{aligned} \qquad (1)$$

Solving first and second equations relatively at α and β, it results:

$$\alpha = \frac{r_2 - 1}{r_2 - r_1}, \quad \beta = \frac{1 - r_1}{r_2 - r_1}$$

Replacing the values of α and β in the 3 and 4th, we obtain, respectively

$$r_1 + r_2 - r_1 r_2 = 2$$
$$- r_1 r_2 (r_1 + r_2) + (r_1 + r_2)^2 - r_1 r_2 = 3$$

Solving this system into $r_1 + r_2$ and $r_1 r_2$ we obtain $r_1 + r_2 = 1$, $r_1 r_2 = -1$; that is r_1 and r_2 must satisfy equation $r^2 - r - 1 = 0$

whose roots are:

$$r = \frac{1 + \sqrt{5}}{2}, \quad r = \frac{1 - \sqrt{5}}{2}$$

Determining α and β from the previous relations, it results for F_i the formula:

$$F_i = \frac{1}{\sqrt{5}} \left[\left(\frac{1 + 5}{2} \right)^{i+1} - \left(\frac{1 - 5}{2} \right)^{i+1} \right]$$

In the algorithm of the golden section, as we have already shown, $r_k = r =$ constant at each iteration, and so we have

$$\frac{x_2^k - a^k}{b^k - a^k} = r \tag{2}$$

$$\frac{b^k - x_1^k}{b^k - a^k} = r \tag{3}$$

To fix the ideas, let us suppose that $f(x_2^k) > f(x_1^k)$

Then $b^{k+1} := x_2^k$, $a^{k+1} := a^k$, $x_2^{k+1} := x_1^k$ (4)

At iteration $k + 1$ the interval will have to decrease in the same relation:

$$\frac{x_2^{k+1} - a^{k+1}}{b^{k+1} - a^{k+1}} = r \tag{5}$$

But, taking into consideration (4), (5) becomes:

$$\frac{x_1^k - a^k}{x_2^k - a^k} = r \tag{6}$$

and (6) can, however, be written in the form

$$\frac{x_1^k - b^k + b^k - a^k}{x_2^k - a^k} = r \tag{6'}$$

Or

$$-\frac{-x_1^k + b^k}{x_2^k - a^k} + \frac{b^k - a^k}{x_2^k - a^k} = r \tag{7}$$

But, dividing (3) by (2) $\dfrac{b^k - x_1^k}{x_2^k - a^k} = 1$ and taking into consideration once again (2) from (7) we infer that r must satisfy:

$$r^2 + r - 1 = 0 \tag{8}$$

Hence results $r = \dfrac{\sqrt{5} - 1}{2} \approx 0.618$ and with it the name of the algorithm.

On the other hand, from (1):

$$x_1^k = a^k + r^2(b^k - a^k), \quad x_2^k = a^k + r(b^k - a^k)$$

so that always $x_i^k < x_2^k$ and the algorithm has a very simple iteration:

(i) If $f(x_2^k) > f(x_1^k)$

$$a^{k+1} := a^k$$

$$b^{k+1} := x_2^k$$

$$x_1^{k+1} = a^{k+1} + r^2(b^{k+1} - a^{k+1})$$

$$x_2^{k+1} = a^{k+1} + r(b^{k+1} - a^{k+1}) = a^k + r(x_2^k - a^k) =$$

$$= a^k + r^2(b^k - a^k) = x_1^k.$$

(ii) If $f(x_2^k) \leqslant f(x_1^k)$

$$a^{k+1} := x_1^k$$

$$b^{k+1} := b^k$$

$$x_1^{k+1} := x_2^k$$

$$x_2^{k+1} = a^{k+1} + r(b^{k+1} - a^{k+1})$$

It can be seen that after n evaluations of the function

$$\delta_n = b^n - a^n = (b^1 - a^1)\, r^{n-1}$$

As regards efficiency, it may be shown that [2, page 15] to obtain a given tolerance, the algorithm of the golden section will not require more than a single evaluation of the function in comparison with the algorithm based on Fibonacci's numbers which, from this point of view, is optimum.

REFERENCES

1. FIACCO, A.V., McCORMICK, G.P.: *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley, N.Y., 1968.
2. KOWALIK, J., OSBORNE, M.R.: *Methods for Unconstrained Optimization Problems*, Elsevier, N.Y., 1968.
3. FLETCHER, R., EDITOR: *Optimization*. Academic Press, London, 1969.
4. ABADIE, J., editor: *Integer and Nonlinear Programming*. North-Holland, 1970.
5. HOOKE, R., JEEVES, T.A.: *Direct Search Solution of Numerical and Statistical Problems*, J. Assoc. Comput, Mach. 8, 1961.
6. BELL, M., PIKE, M.C.: *Remark on Algorithm 178 (E4) Direct Search*, Comm. ACM, 9, 1966.
7. ROSENBROCK, H.H.: *An Automatic Method for Finding the Greatest or Least Value of a Function*, Computer J., 3, 1960.
8. DAVIES, D.: *Some Practical Methods of Optimization*, an article included in [4].
9. FLETCHER, R.: *Function Minimization without Evaluating Derivatives — A Review*, Computer J., 8 (1965).
10. POWELL, M.J.D.: *An Efficient Method of Finding the Minimum of a Function of Several Variables Without Calculating Derivatives*, The Computer Journal, Vol 7, 1964.
11. WOLFE, P.: *Convergence Theory in Nonlinear Programming*, in [4].
12. * * *: *Discussions and Panel-Discussion at Keele Conference*, in [3].
13. SMITH, C.S.: *The Automatic Computation of Maximum Likelihood Estimates*, N.C.B. Scientific Dept. Report S.C. 846/MR/40(1962).
14. SPENDLEY, W., HEXT G.R. HIMSWORTH, F.R.: *Sequential Application of Simplex Designs in Optimization and Evolutionary Operation*, Technometrix, 4, 1962.
15. NELDER, J.A., MEAD, R.: *A Simplex Method for Function Minimization*, Computer J., 7, 1965.
16. DAVIES, D., SWANN, W.H.: *Review of Constrained Optimization*, in [3].
17. BOX, M. J.: *A Comparison of Several Current Optimization Methods and the Use of Transformations in Constrained Problems:* Computer J., 9, 1969.
18. BOX, M.J., DAVIES, D., SWANN, W.H.: *Nonlinear Optimization Techniques*, I.C.I. Monograph on Mathematics and Statistics, Oliver and Boyd, Edinburgh, 1969.
19. ARROW, K.J., HURWICZ, L., UZAWA, H.: *Studies in Linear and Nonlinear Programming*, Stanford University Press, Stanford, 1958.
20. FLETCHER, R.: *A Review of Methods for Unconstrained Optimization*, in [3].
21. DRAGOMIRESCU, M., MALIȚA, M.: *Programarea Pătratică (Quadratic Programming)*. Editura științifică, Bucharest, 1968.
22. SWANN, W.H.: *Report on the Development of a New Direct Search Method of Optimization*, I.C.I.Ltd. Central Instrument Laboratory Research Note 64/3(1964).
23. BOX, M.J.: *A New Method of Constrained Optimization and a Comparison with Other Methods*, Computer J., 8, 1965.
24. AMUNATEGUI, J., DUMITRU V.: *Computer Experiments with an Unconstrained Nonlinear Program.* Economic Computation and Economic Cybernetics Studies and Research No. 3, 1970.
25. KIEFER, J.: *Sequential Minimax Search for a Minimum.* Proc. Amer. Math. Soc., 4, 1953.
26. JOHNSON, S.M.: *Best Exploration for Minimax is Fibonaccian.* The RAND Corporation, RM -1590, 1955.

METHODE D'OPTIMISER LES FONCTIONS, SANS EVALUATION DE LA DERIVE

Cet article constitue un aperçu général sur quelques méthodes et techniques utilisées pour optimiser les fonctions sans l'emploi des dérivées, en offrant certains algorithmes destinés à faciliter les tâches de la programmation.

La seconde partie de l'article renfermera des organigrammes, des subroutines, ainsi que des résultats numériques.

МЕТОДЫ ОПТИМИЗАЦИИ ФУНКЦИЙ БЕЗ ОЦЕНКИ ПРОИЗВОДНОЙ

Настоящая статья представляет собой общий обзор методов и техники оптимизации функций без использования производных с попыткой представления алгоритмов таким образом, чтобы облегчить задания программирования.

Во второй части будут представлены органиграммы, субрутины электронной вычислительной машины и численные результаты.