J. ABADIE

Optimization Problems with Coupled Blocks*

I. INTRODUCTION

In what follows, we consider a graph, or network, $\mathbf{G}$, which can be, depending on the problem under examination, directed or not. Unless otherwise specified, however, $\mathbf{G}$ will be undirected.

To each node I is attached a set of unknowns x_j, $j \in P(I)$, which form the vector $x_{P(I)}$ hereafter called *proper to node I*. To two distinct nodes are attached two proper vectors which have no common component:

$$I \neq J \Rightarrow P(I) \cap P(J) = \emptyset.$$

To node I is adjacent a set of nodes, each of them being said *linked to node I*; let $L(I)$ be the set of such nodes, and $x_{L(I)}$ the vector the components of which are x_j, $j \in L(I)$,; the latter vector is called: *linked* to node I.

To each node I is attached a set of costraints, which will be conveniently written:

$$f_i(x_{P(I)}, x_{L(I)}) = 0, \; i \in I,$$

or, in more compact form:

$$f_I(x_{P(I)}, x_{L(I)}) = 0, \tag{1.1}$$

We see thus that node I is identified which a set of indices of (scalar) constraints, and that these constraints do not explicitly contain variables other than those which are proper or linked to node I. The set of constraints attached to node I is also called *proper to node I*.

---

* Invited communication, Symposium for National Economy Planning, Novosibirsk, U.S.S.R., 22–27 June, 1970.

If some constraint, say

$$f_i(x_{P(I)},\ x_{L(I)}) \leqslant 0,$$

is an inequality, a slack variable is added to obtain the equivalent constraints:

$$x_i + f_i(x_{P(I)},\ x_{L(I)}) = 0.$$

$$x_i \geqslant 0.$$

The slack-variable x_i may be considered either as part of the proper vector $x_{P(I)}$, or as the proper vector attached to a supplementary node linked to node I only.

We assume all variables bounded above and below:

$$a_j \leqslant x_j \leqslant b_j, \tag{1.2}$$

for all index j under consideration. The latter constraints, called *trivial constraints*, are not treated via slack-variables.

Examples 1. In the Electrical Power Industry, the power flow obeys two equations per node:

$$C_i - P_i + G_i V_i^2 + V_i \sum_{j \in L(i)} G_{ij} V_j \cos(\theta_i - \theta_j - \varphi_{ij}) = 0$$

$$D_i - Q_i + H_i V_i^2 + V_i \sum_{j \in L(i)} H_{ij} V_j \sin(\theta_i - \theta_j - \varphi_{ij}) = 0,$$

where the proper vector at node i is $(P_i,\ Q_i,\ V_i,\ \theta_i)$, respectively: active and reactive power production at node i, voltage magnitude and phase angle.

2. In an Economy divided into sectors (for instance, geographical), the system of equations (1) describes, together with the corresponding graph, **G**, the direct interrelationships between the various sectors.

3. In discrete optimal control we may have a system of the following form:

$$f_t(x_t, x_{t+1}, u_t, u_{t+1}) = 0,\ t = 0,\ \ldots,\ T-1.$$

The graph **G** here is identified with a chain having nodes $t = 0, \ldots, T$. Here x_t is the (vector) state variables, u_t the (vector) control variable at time t: they form the vector proper to time t. As simpler forms, one can have

$$f_t(x_t, x_{t+1}, u_t) = 0 \tag{1.3}$$

or

$$x_{t+1} - f_t(x_t, u_t) = 0$$

More complicated forms are also possible, when the description of the physical system involves interaction at instants separated by more than one unit of time. The graph **G**, then, is no more a chain, and should more appropriately be a *directed* graph, as it should already be in the last two simpler forms written above.

As a special case of (1.3), one finds the important problem, in Mathematical Economics, of maximizing the sum of the utilities, over the periods 0, 1,..., T in an expanding economy (of course, in the case of a final utility function at time T, some use should be done of turnpike theorems).

4. The graph G can be the product of the one in Example 2 and the one in Example 3: the system of equations in Example 2 describes the system at period t; we imagine that the equations (1) or the bounds (2) contain state parameters p_t, which in turn are governed by control variables u_t. Thus, variables x_t, p_t enters (1), (2), (where f_t depends on t), while variables p_t, u_t enters (3). Such an example is related for instance, to investments problem in a growing Economy, over a period of time $(0, T)$.

5. It happens sometimes that some variables are, in a natural way, attached to an edge of the graph, G, instead of any of the two adjacent nodes. This would be the case, may be, for transportation, to take just one instance. The general case however applies, by creating a node (I, J) for transportation between nodes I and J. To take another example from the Electrical Power Industry, the inequality

$$-\alpha_{ij} \leqslant \theta_i - \theta_j \leqslant \alpha_{ij}$$

is transformed into:

$$\theta_i - \theta_j + w_{ij} = 0$$

$$-\alpha_{ij} \leqslant w_{ij} \leqslant \alpha_{ij},$$

where w_{ij} is the slack variable. Thus, we treat w_{ij} as being the (unique) variable proper to some fictitious node (i, j) created for that purpose. The node (i, j) may conveniently be placed in the middle of the edge (i, j), which is thus divided into two new edges. It may also be identified with the edge (i, j) itself: the first possibility is chosen for simplicity in the terminology.

Let us be given some objective function $\alpha(x)$, which we assume continuously differentiable. Our aim is to numerically solve the following optimization problem:

minimize $\varphi(x)$

subject to:

$$f(x) = 0$$

$$a \leqslant x \leqslant b,$$

where the constraints and variables are attached to a graph G in the manner described above.

2. METHODS OF TANGENT DIRECTIONS

Let us consider the following mathematical programming problem:

minimize $\varphi(x)$

subject to

$$f(x) = 0 \tag{2.1}$$

$$a \leqslant x \leqslant b, \tag{2.2}$$

where x, a, b are n-dimensional column-vectors, $f(x)$ is, for any x under consideration, an m-dimensional column-vector, and, likewise, $\varphi(x)$ is a real number. Both φ and f are assumed continuously differentiable on the paralle-

lotope P defined by (2). By allowing some of the a_j or b_j to be $-\infty$ or $+\infty$, any mathematical programming problem can be written in the preceding standard form. However, we shall assume a and b bounded, which can always be done in practice, though some of the a_j, b_j can be large.

The notation $\varphi'(x)$ designates the n-dimensional row-vector, the components of which are the partial derivatives $\frac{\partial \varphi}{\partial x_j}$. Likewise, $f'(x)$ is the (m, n)-matrix the elements of which are $\frac{\partial f_i}{\partial x_j}$ (the Jacobian of constraints (1)). We assume that the rank of $f'(x)$ is m at all points under consideration.

Let x° be a feasible point, that is a point satisfying (1), (2). A *tangent direction* at x° is an n-dimensional column-vector ξ satisfying:

$$f'(x^\circ)\,\xi = 0 \tag{2.3}$$

$$\xi_j \begin{cases} \geqslant 0 \text{ if } x_j^0 = a_j \\ \leqslant 0 \text{ if } x_j^0 = b_j. \end{cases} \tag{2.4}$$

An *improving* tangent direction is a tangent direction satisfying, in addition to (3), (4), the following inequality:

$$\varphi'(x^\circ)\,\xi < 0. \tag{2.5}$$

If no improving tangent direction exists, then x° satisfies the Kuhn-Tucker optimality condition (we say that x° is *stationary*). In any case, linear programming techniques permit, in a finite number of steps, either to find an improving tangent direction, or to conclude that none exists: it suffices to minimize $\varphi'(x^\circ)$, ξ, subject to (3), (4) and $-M \leqslant \xi_j \leqslant M$, $\forall j$, and to stop the calculations after having obtained a non-zero value of the objective function.

Once an improving tangent direction ξ is available, one can say that, for small enough positive values of the real number θ:

$$\varphi(x^\circ + \theta\xi) < \varphi(x^\circ)$$

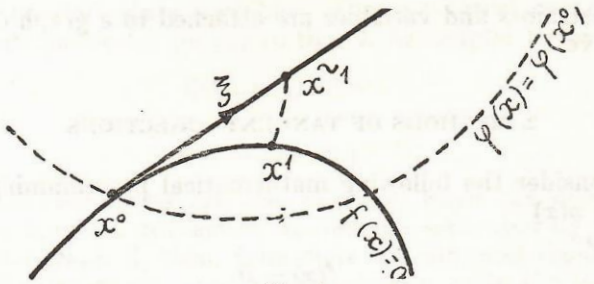


Fig. 1

Let θ_1 be some positive value of θ, and $\tilde{x}^1$ be the point:

$$\tilde{x}^1 = x^\circ + \theta_1\,\xi.$$

$\tilde{x}^1$ will not generally satisfy (1), (2).

If it does not satisfy (2), we can either replace θ_1 by a smaller value such, that:

$$x \leqslant x^\circ + \theta\xi \leqslant b,$$

or keep θ_1 as it is and project $\tilde{x}^1$ onto the parallelotope P, that is replace for any index j, $\tilde{x}_j^1$ by its nearest bound a_j, b_j if it does not lie in the interval (a_j, b_j).

If $\tilde{x}^1$ satisfies (2) and not (1), then a solution of (1) should be computed using some iterative method, starting from $\tilde{x}^1$. Assume this method converges if θ is small enough (here again, it may be necessary to diminish θ if no or slow convergence is observed), and call x^1 the solution obtained. If $\varphi(x^1) < \varphi(x^\circ)$, then x^1 replaces x° and the process is repeated; otherwise, θ is again diminished, and another feasible point x^1 is computed.

In the case where the original problem contains inequality constraints

$$f_i(x) \leqslant 0,$$

only those which are binding at the point (x°, or x^1, or x^1) under consideration need receiving the treatment with slack variables. Nevertheless, in what follows, we assume for simplicity that all non-trivial inequalities have been transformed into equations by mean of slack variables.

The method of tangent directions includes the method of feasible direction, of Zoutendijk [13]. It also includes other methods, such as the projected gradient method for nonlinear constraints, of J.B. ROSEN [11].

We now describe, as a subclass of the method of tangent directions, the Generalized Reduced Gradient method, hereafter abbreviated as "GRG" method, of J. Carpentier and the Author.

3. THE GRG METHOD

We consider again the problem: minimize $\varphi(x)$
subject to:

$$f(x) = 0 \tag{3.1}$$

$$a \leqslant x \leqslant b. \tag{3.2}$$

Let a feasible point x° be given in such a way that there exists a decomposition of the n-dimensional vector x into an m-dimensional vector x_B and an $(n-m)$-dimensional vector x_N such that the following *non-degeneracy assumption* holds:

$$a_j < x_j^0 < b_j, \ \forall j \in B \tag{3.3}$$

$$\frac{\partial f}{\partial x_B^0} \text{ is non-singular,} \tag{3.4}$$

where $\dfrac{\partial f}{\partial x_B^0}$ is the (m, m)-matrix of partial derivatives $\dfrac{\partial f_i}{\partial x_j}$, $i = 1, \ldots, m$; $j \in B$, computed at x^0.

Let us consider, by the implicit function theorem, x_B as a function of x_N, defined, continuous, and even continuously differentiable, in some neighbourhood of x_N^0:

$$x_B = F(x_N), \tag{3.5}$$

such that (1) is identitically satisfied. Substituting (5) into the objective function $\varphi(x)$, we get a function:

$$\Phi(x_N) = \varphi(x_N, F(x_N)), \tag{3.6}$$

the gradient of which is, using (1), (5), and (6):

$$\Phi'(x_N) = \frac{\partial \varphi}{\partial x_N} - \frac{\partial \varphi}{\partial x_B}\left(\frac{\partial f}{\partial x_B}\right)^{-1} \frac{\partial f}{\partial x_N}. \tag{3.7}$$

Though there is no way, in general, to have an explicit formula for (6), formula (7) is easily applicable at $x^0 \cdot \Phi'(x_N^0)$ is called the *reduced gradient* at x^0, and is denoted by g^0.

We define now the *projected reduced gradient* as:

$$\forall j \in N : p_j^0 = \begin{cases} 0 \text{ if } x_j^0 = a_j \text{ and } g_j^0 > 0 \\ 0 \text{ if } x_j^0 = b_j \text{ and } g_j^0 < 0 \\ g_j^0 \text{ otherwise.} \end{cases} \tag{3.8}$$

Setting $\xi_N = -p_N^0$, we have

$$\Phi'(x_N^0)\, \xi_N < 0 \tag{3.9}$$

(unless $p_N^0 = 0$, in which case x^0 is stationary).

We can now easily compute ξ_B such that $\xi = (\xi_B, \xi_N)$ is a tangent vector. The condition:

$$\frac{\partial f}{\partial x_B^0}\, \xi_B + \frac{\partial f}{\partial x_N^0}\, \xi = 0 \tag{3.10}$$

gives

$$\xi_B = -\left(\frac{\partial f}{\partial x_B^0}\right)^{-1} \frac{\partial f}{\partial x_N^0}\, \xi_N. \tag{3.11}$$

Since

$$\varphi'(x^0)\, \xi = \Phi'(x_N^0)\, \xi_N,$$

we see from (9) that the tangent direction ξ is also an improving direction.

Having chosen some positive value θ_1, the next step in the method of tangent directions is to compute

$$\tilde{x}_B^1 = x_B^0 + \theta_1 \xi_B$$

$$\tilde{x}_N^1 = x_N^0 + \theta_1 \xi_N,$$

and to apply some iterative method, starting from the (generally) non-feasible point $\tilde{x}^1$, to compute a feasible point x^1.

In the GRG method, this is done by Newton's method or one of its modifications, in order to compute x_B^1, while we keep x_N at its value $\tilde{x}_N^1$. The modification we currently use is the following, starting with $\tilde{x}_B^1$:

$$\tilde{x}_B^{t+1} = \tilde{x}_B^t - \left(\frac{\partial f}{\partial x_B^0}\right)^{-1} f(\tilde{x}_B^t, \tilde{x}_N^1), \tag{3.12}$$

where the (m, m) — matrix $\dfrac{\partial f}{\partial x_B^0}$ is the same as the one used in (7) and (11), that is, computed at x°, instead of $(\tilde{x}_B^t, \tilde{x}_N^1)$ as it should be in Newton's method.

Figure 2 illustrates this part of the computation, in the case where all iterates $(\tilde{x}_B^t, \tilde{x}_N^1)$ belong to the parallelotope $a_j \leqslant x_j \leqslant b_j$, $\forall j$. The other

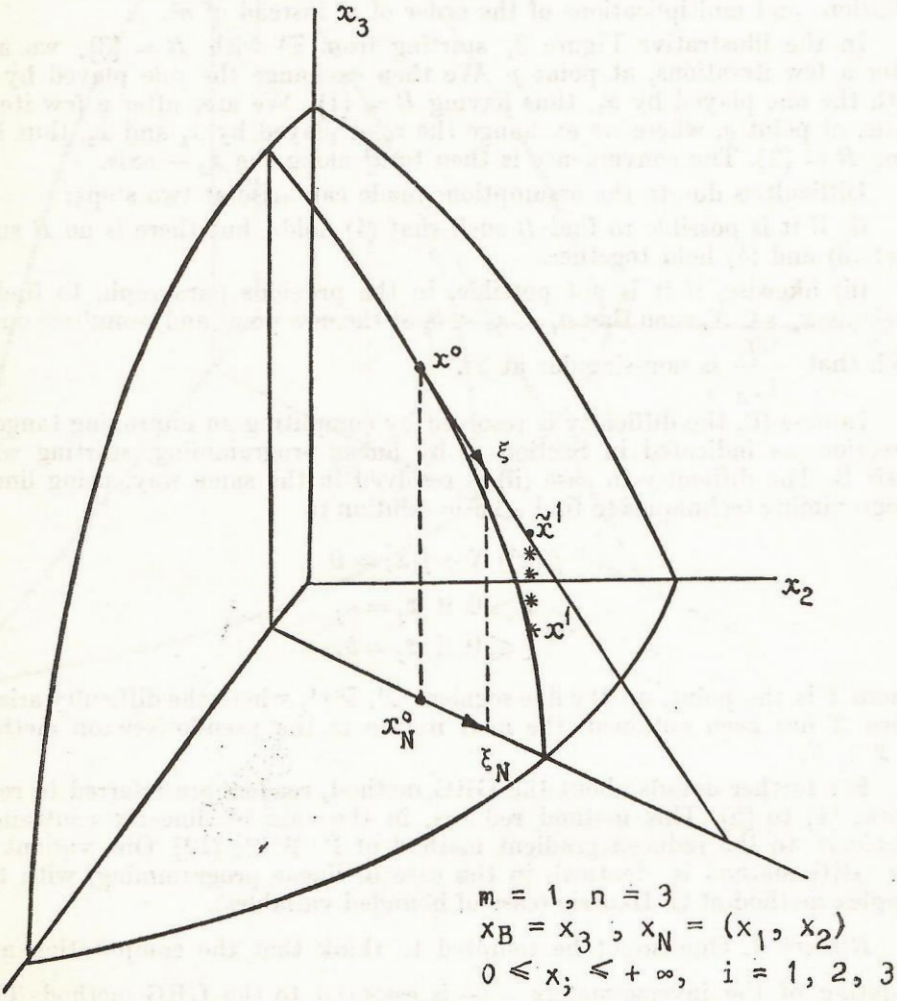


Fig. 2

case is illustrated in Figure 3, where we assume that, for some t, $\tilde{x}_B^t$ satisfies $a_B \leqslant x_B \leqslant b_B$, while this is not true for $\tilde{x}_B^{t+1}$. We compute then the intersection of the line segment joining $\tilde{x}_t^B$, $\tilde{x}_B^{t+1}$ and of the boundary of the parallelotope. For this new point, we have $x_r = a_r$ or b_r, for some $r \in B$. We exchange the role of x_r with the role of x_s, for some $s \in N$, chosen in such a way that $a_s < x_s < b_s$ at this new point, and that the matrix $\dfrac{\partial f}{\partial x_B}$ is non-singular at x^0 for the new index set B (provisions are made to still continue the computations if, exceptionnally, the latter conditions do not give any index s). The computation of the new $\left(\dfrac{\partial f}{\partial x_B^0}\right)^1$ is made, as in the revised simplex method, by a pivotal operation instead of an inversion; this takes a number of additions and multiplications of the order of n^2 instead of n^3.

In the illustrative Figure 3, starting from $\tilde{x}^1$ with $B = \{3\}$, we are, after a few iterations, at point p. We then exchange the role played by x_3 with the one played by x_1, thus having $B = \{1\}$. We are, after a few iterations, at point q, where we exchange the roles played by x_1 and x_2, thus having $B = \{2\}$. The convergence is then to x^1 along the x_2 — axis.

Difficulties due to the assumptions made can arise at two steps:

(i) if it is possible to find B such that (4) holds, but there is no B such that (3) and (4) hold together.

(ii) likewise, if it is not possible, in the previous paragraph, to find a variable x_s, $s \in N$, such that $a_s < x_s < b_s$ at the new point and, simultaneously such that $\dfrac{\partial f}{\partial x_B}$ is non-singular at x°.

In case (i), the difficulty is resolved by computing an improving tangent direction, as indicated in Section 2, by linear programming, starting with basis B. The difficulty in case (ii) is resolved in the same way, using linear programming techniques to find a basic solution to

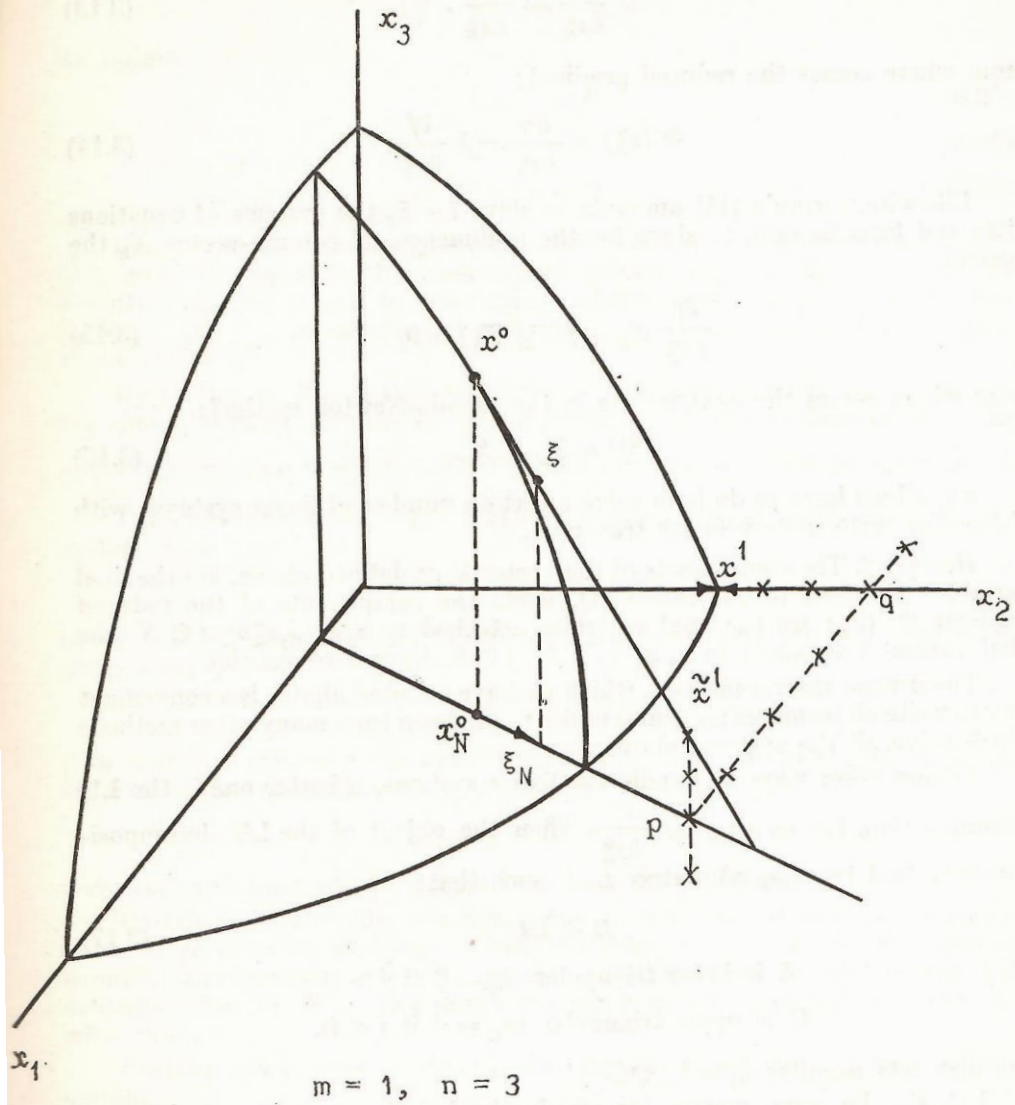
$$f'(x^\circ)\, X + f(\bar{x}) = 0$$

$$X_j \begin{cases} \geqslant 0 \text{ if } \bar{x}_j = a_j \\ \leqslant 0 \text{ if } \bar{x}_j = b_j, \end{cases}$$

where $\bar{x}$ is the point, on the line segment $\tilde{x}^t$, $\tilde{x}^{t+1}$, where the difficulty arises. Once X has been obtained, the next iterate in the pseudo-Newton method is $\bar{x} + X$.

For further details about the GRG method, readers are referred to references [1] to [5]. This method reduces, in the case of linearly contrained problems, to the reduced gradient method of P. Wolfe [12]. One variant of the GRG method is identical, in the case of linear programming, with the simplex method of G. Dantzig (case of bounded variables).

Remark 1. One might be tempted to think that the computation and updating of the inverse-matrix $\dfrac{\partial f}{\partial x_B^0}$ is essential to the GRG method. This



$m = 1, \quad n = 3$

$0 \leqslant x_j \leqslant +\infty, \quad j = 1, 2, 3$

Fig. 3

be a matrix for which we wish the LU decomposition. Setting

$$B_{(2)} = \begin{pmatrix} B_{(1)} & P \\ Q & R \end{pmatrix} = \begin{pmatrix} L_{(1)} & 0 \\ Q' & L' \end{pmatrix} \begin{pmatrix} U_{(1)} & P' \\ 0 & U' \end{pmatrix} \tag{3.19}$$

we obtain:

$$L_{(1)} P' = P \tag{3.20}$$

$$Q' U_{(1)} = Q \tag{3.21}$$

$$L' U' = R - Q' P'. \tag{3.22}$$

Let us consider the case where P has one column, and Q has one row. Then (20) gives in sequence the elements of column P', say P'_1, P'_2, P'_3, ... (P'_1 directly and the others by successive substitutions). Likewise (21) gives Q'_m, Q'_{m-1}, Q'_{m-2}, ... in sequence. Finally, (22) gives the scalars L', U', provided $R \neq Q'P'$.

By taking, as $B_{(1)}$, nested submatrices of successive ranks 1, 2, ..., $m-1$, the above formulae give the LU decomposition of B in a number of additions and multiplications of $\frac{1}{3} m^3 + 0\ (m^2)$, instead of $m^3 + 0\ (m^2)$ for inverting B

The knowledge of $B = LU$ permits the computation of the solution of a system such as

$$By = c$$

in $m^2 + 0\ (m)$ additions and multiplications, by solving the two systems in succession:

$$Lz = c,\ Uy = z$$

(this is done in the same way as (20) and (21) has been solved above). It should be noticed that the formula

$$y = B^{-1} c$$

gives the same amount of computations, once B^{-1} is known.

Bartels and Golub [6] have shown how to compute the new LU decomposition of B, after an exchange of two variables, x_r, x_s is made. Fortunately enough, the number of additions and multiplications to update the LU decomposition of B is essentially the same as for updating B^{-1}, that is: $m^2 + 0(m)$.

Further advantages of the LU decomposition will be seen in the next section.

4. APPLICATION OF THE GRG METHOD

We explain in this section how to apply the GRG method to the problem discussed in the introduction. We assume that this problem is a large-scale one, and that the average number of edges incident with a node is small,

compared to the number of couples of nodes (connected or not) in graph G. Figure 4 illustrates such a case for a very graph small having 10 nodes and 13 edges. The number of couples of nodes is 45. The ratio 13/45 is not very small here, but in larger problems it could be very much smaller: for instance, if G possesses N nodes, each of which is linked to 6 others, then G possesses $3N$ edges, as compared with $N(N-1)/2$ pairs of nodes, which gives the ratio $6/(N-1)$, which is below 1% if N greater than 600.

Assume for simplicity that, associated with graph, G, there is a system of 10 linear equations in 10 unknowns (one proper equation and one proper unknown per node). In figures 5 and 6, the matrix B of the system is shown, with dots corresponding to the non-zero elements, with different orderings of rows lumns. When LU decomposition is made, the non-zero elements in the upper-diagonal part of B are transformed into generally non-zero elements of U, while some other non-zero elements are created in U (crosses in figures 5 and 6). The same is true for L, where due to the symmetric appearance of B, the dots and crosses are symmetric to those of U.

Counting only the non-zero off-diagonal entries of U, we see that the ratio of created non-zero elements versus the non-zero elements coming from those of B is 3/13 in Figure 5, and 21/13 in Figure 6. We then see how strong is the dependance on the orderings of rows and columns.

It is not known how to find an optimal ordering. The one in Figure 5 is obtained by the following empirical rules, in which S, N, $L(i)$, $L'(i)$ are sets of nodes, and $!\,L(i)\,!$ is the number of elements in $L(i)$.

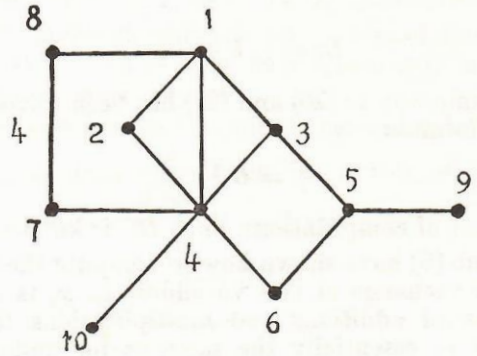


Fig. 4

1. Start with: $L'(i) = L(i)$, $\forall i$; $S = N = \emptyset$.

2. Among nodes $i \notin S$, select one for which $|\,L'(i)\,|$ is minimum. In case of tie, choose $i \notin N$. In case there remain ties, choose i such that $|\,L(i)\,|$ is minimum. In case some ties still remain, pick any i among the candidates which survive. Augment set S with element i. Augment set N with element i and with $L'(i)$. For each $j \notin S$, substract from set $L'(j)$ all elements in $L'(i)$.

Figure 6 is obtained by reversing the order given by the preceding rules.

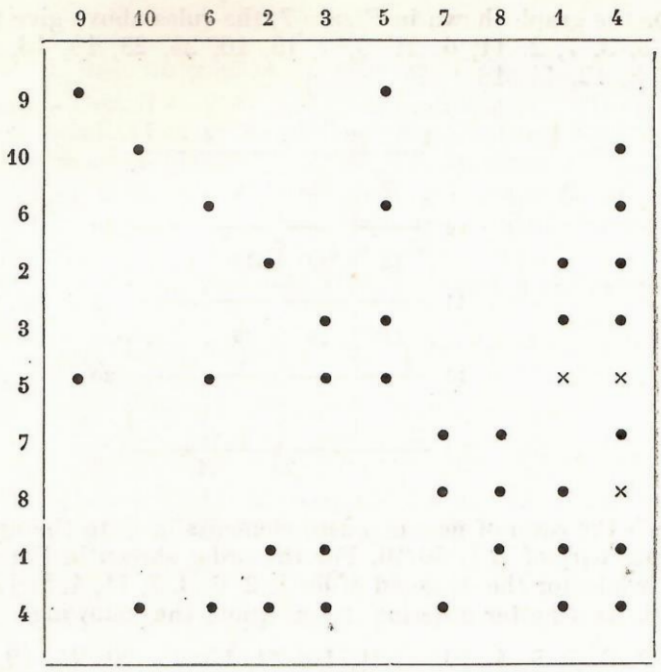


Fig. 5
ratio: 3/13

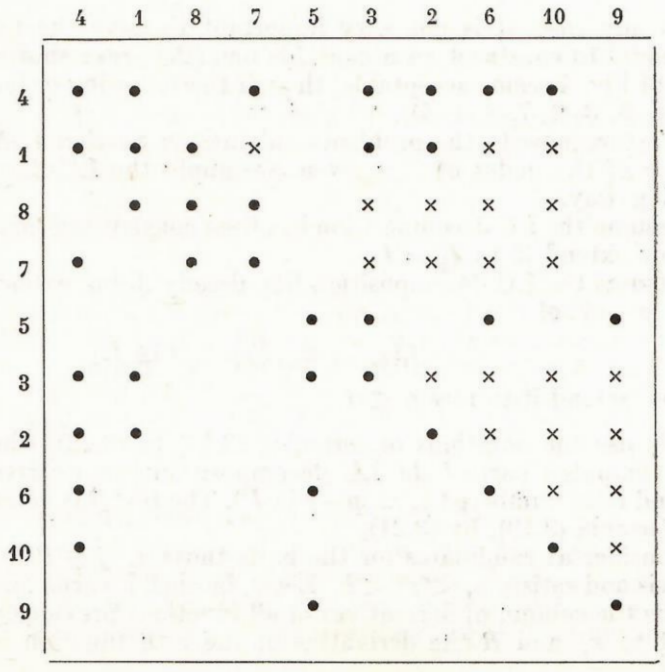


Fig. 6
ratio: 21/13

For the graph shown in Figure 7, the rules above give the order:
1, 5, 3, 7, 2, 11, 6, 21, 9, 4, 15, 10, 25, 23, 19, 13, 17, 16, 20, 22, 24, 8, 12, 14, 18

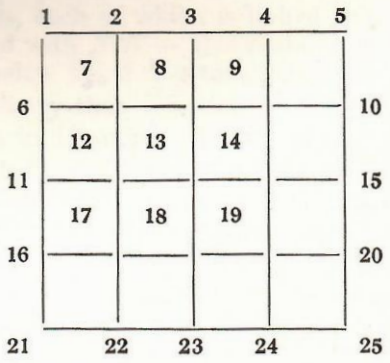


Fig. 7

for which the ratio of new non-zero elements in U to the non-zero element in the upper part of B is 46/40. For the order shown in Figure 7, this ratio is 64/40, while for the diagonal order 1, 2, 6, 3, 7, 11, 4, 8, 12, 16, ..., the ratio is 50/40. As a better ordering, let us quote the following:

1, 2, 6, 7, 5, 4, 10, 9, 21, 16, 22, 17, 25, 20, 24, 19, 3, 8, 11, 12, 15, 14, 23, 18, 13,

for which the ratio is 39/40.

In any case, it is not very important to have the best ordering: it is only useful to construct an acceptable one (the order shown in Fig. 5 for the graph of Fig. 4 seems acceptable, though the following order gives ratio 2/13: 9, 10, 6, 5, 3, 2, 7, 8, 1, 4).

We turn now to the problem explained in Section 1. Assume that some ordering of the nodes of G is given, we apply the LU decomposition in the following way.

Assume the LU decomposition has been constructed for nodes $I_1, ..., I_{q-1}$. We now extend it to $I_q = I$.

Assume the LU decomposition has already been extended to equations 1, ..., p − 1 of

$$f_i(x_{P(I)}, x_{L(I)}) = 0, \; i \in I.$$

We now extend it to row $p \in I$.

We use the notations of formulae (3.17) to (3.22), where $B_{(1)} = L_{(1)}U_{(1)}$ is the computed part of the LU decomposition (rows corresponding to $I_1,...,$ I_{q-1} and rows numbered 1, ..., $p-1$ in I_q). The row Q is determined, hence also Q' in formula (3.19), by (3.21).

Consider as candidates for the basis those x_j, $j \in P(I)$, which are not in the basis and satisfy $a_j < x_j^0 < b_j$. Let x_s be such a variable. In formula (3.19), let P be the column of derivatives of all functions previously considered, with respect to x_s, and R the derivative of the p-th function in f_I, with respect

to x_s. We now compute P' by (3.20) and $R - Q'P'$ by (3.22). If $R - Q'P' \neq \neq 0$, then the LU decomposition is extended to equations 1, ..., p of node I. If $R - Q'P' = 0$, then we consider another candidate x_s.

It might happen that $R - Q'P' = 0$ for all candidates. This case will be examined in the Appendix. Let us assume here, for simplicity, that the operations described can be accomplished up to the last equation.

In this way, very large problems attached to a graph G can be solved entirely in the central memories of the big machines available to-day. In any case, a substantial amount of computations is saved.

Let us now turn to those discrete optimal control problems where the constraints are:

$$x^{t+1} = f_t(x^t, u^t), \quad t = 0, ..., T-1 \tag{4.1}$$

$$a^t \leqslant x^t \leqslant b^t, \quad t = 0, ..., T \tag{4.2}$$

$$\alpha^t \leqslant u^t \leqslant \beta^t, \quad t = 0, ..., T-1. \tag{4.3}$$

More general problems could be examined, in the same way as in the next paragraphs, for instance: (1) could be replaced by:

$$f_t(x^{t+1}, x^t, u^t) = 0$$

or by

$$f_t(x^{t+1}, x^t, u^{t+1}, u^t) = 0;$$

some constraints of the types:

$$F_t(x^t) = 0, \; F_t(u^t) = 0, \; F_t(x^t, u^t) = 0,$$

might be added. For simplicity in the discussion, we confine ourself to (1), (2), (3).

The jacobian of (1) has the shape shown in Figure, 8, where the non-zero elements are only in the exhibited unit submatrices, and in the submatrices $H_0, ..., H_{T-1}$; $K_0, ..., K_{T-1}$, and where:

$$H_t = -\frac{\partial f_t}{\partial x^t}, \quad K_t = -\frac{\partial f_t}{\partial u^t}$$

The basis matrix is obtained by putting in the basis as many 1's (from the unit submatrices) as we can, i.e.: as many as there are state variables x_j^t satisfying, for the current feasible point, the constraints (2) as strict inequalities $a_j^t < x_j^t < b_j^t$. For the others, x_j^t is replaced in the basis by some u_i^{t-1} chosen in such a way that:

(i) $\alpha_i^t < u_i^t < \beta_i^t$

(ii) the corresponding basis matrix is non-singular.

The structure of the corresponding basis matrix is shown in Figure 9, where the shaded areas correspond to submatrices of H_t, K_t.

The agreeable circumstance in such a basis matrix is that *we only need to produce the LU decomposition of the square block-diagonal submatrices.* More-

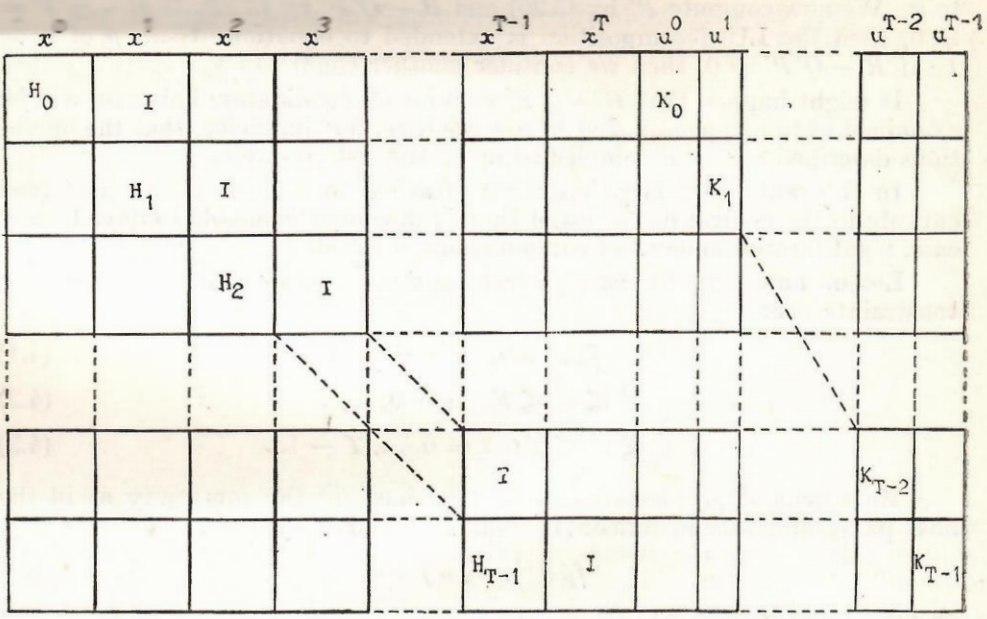

x⁰ x¹ x² x³ x^{T-1} x^T u⁰ u¹ u^{T-2} u^{T-1}
H0
I
K0
H1
I
K1
H2
I
I
K_{T-2}
H_{T-1}
I
K_{T-1}

Fig. 8

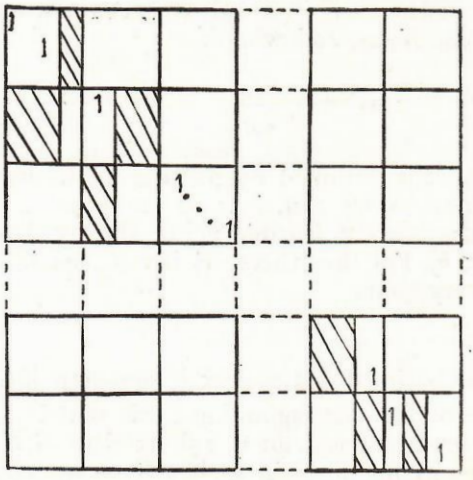

1
1
1
1
1
1
1

Fig. 9

over, in each of the latter submatrices, we need only producing the LU decomposition of that part which does not correspond to unit columns.

For further details, readers are referred to [1].

5. NUMERICAL ADVANCES

We present in this section the present state of numerical experiments along the preceding lines. We are not close to achievement. Nevertheless, it will be seen that the success for solving large optimization problems attached to a graph G should be within reach.

1. The GRG method

Numerical experiments are reported in [5]. The evidence therein displayed show that the method is capable of accuracy (relative errors of 10—8 to 10—12 are the common case). Concerning speed, the Reader is referred to [7] where Colville, making a classification of some 30 numerical methods by way of experimenting them on machines, put GRG as the leading method.

The largest problem tried (and solved) using the GRG method is small, comparatively to those we are examining in this paper: it involves 50 quadratic inequalities in 50 unknowns, and a quadratic objective function. Time is not short: about 4 minutes on a CDC 6600. On the other hand, a lot of computing time is used in inverting, at each iteration, a 50×50 matrix. It can be shown that, using some of the devices indicated in [1], the corresponding computing time could be divided by 20 (not much more, because the 50×50 matrix in this problem, is not attached to a graph: the jacobian indeed has *no* zero element).

2. Computations in the Electrical Power Industry

We refer here to the problem quoted in Section 1. Such problems are currently solved in some Companies (Electricité de France, Bonneville Power Administration), using various methods. The most powerful code seems to be the one produced at the Bonneville Power Administration, by Tinney and co-workers. They use a method which is very similar to the GRG method *plus* penalties for a few constraints (in his GRG method, the choice for variables exchanges is restricted; the penalty is of the Courant type: for some of the constraints $a_j \leqslant x_j \leqslant b_j$, designated in advance, a penalty $M(x_j - a_j)^2$ or $M(x_j - b_j)^2$ is introduced, where M tends to $+\infty$). Tinney uses some ordering device, followed by the LU decomposition, and solves, in typical instances, problems with 450 nodes (i. e.: 900 nonlinear equations), 700 branches, in 1 to 3 minutes on a CDC 6400. Producing a LU decomposition requires 2 seconds, and solving a linear systems using this decomposition requires 1/2 second.

3. Present possibilities.

It is already very realistic to consider that solving large scale nonlinear optimization problems of the types mentioned in Section 1 of the present paper is feasible, and is mainly a question of will, and investment in the programming work.

Using two basis matrices in the LU decompositions

The LU decomposition in Section 4 was made assuming that there always exists, among the variables proper to node I and such that $a_j < x_j^0 < b_j$, some for which $R - Q'P' \neq 0$.

In the contrary case, the general principle underlying GRG leads us to consider a basis which is not of the type described above. In ordrer no to loose the efficiency of the ordering, we can use *two* basis matrices, one of which, say B', is the basis according to the GRG method, and the other, say B, is the basis attached to graph G. For instance, assume the jacobian is:

	1	2	3	4
I_1	2	−1	−1	
I_2	−1	1/2	2	−1
I_3		−1	−1	2

which three nodes I_1, I_2, I_3, and $P(I_1) = \{1\}$, $P(I_2) = \{2, 3\}$, $P(I_3) = \{4\}$. Assume furthermore that $0 \leqslant x_j \leqslant 1$, $j = 1, \ldots, 4$, defines the parallelotope, and that $x_1^0 = 1/2$, $x_2^0 = 1/2$, $x_3^0 = 0$, $x_4^0 = 1/2$. The basis for the GRG method is the matrix

$B' =$

2	−1	
−1	1/2	−1
	−1	2

the determinant of which is −2. Nevertheless, we cannot apply the method of section 4, because $R - Q'P' = 0$ at the second step (this comes from the fact that the submatrix of B' formed by the first two rows and columns is singular). We choose B as:

$B =$

2	−1	
−1	2	−1
	−1	2

All we need now is a simple relationship between B'^{-1} and B^{-1}.

Let us turn to two matrice B, B', where $\beta = B^{-1}$ is assumed known, and where B' is obtained from B by replacing a set of columns A by an equal number of new columns A'. Let $\beta_{(1)}$ be the submatrix of β formed by those rows of β which have the same indices as the columns of B which form A, and $\beta_{(2)}$ be the submatrix of β formed by the other rows. It is a straightforward matter to check the following formula:

$$B'^{-1} = \begin{bmatrix} (\beta_{(1)} A')^{-1} & 0 \\ -\beta_{(2)} A' (\beta_{(1)} A')^{-1} & I \end{bmatrix} \begin{bmatrix} \beta_{(1)} \\ \beta_{(2)} \end{bmatrix},$$

which will be written as:

$$B'^{-1} = E B^{-1}.$$

All the need to compute, in order to apply the formula, is $(\beta_{(1)} A')^{-1}$ and $\beta_{(2)} A'$. If V is a column vector, then $W = B'^{-1} V$ is computed by the following steps:

1. $\beta V = \begin{bmatrix} V'_{(1)} \\ V'_{(2)} \end{bmatrix}$
2. $\beta A' = \begin{bmatrix} \beta_{(1)} A' \\ \beta_{(2)} A' \end{bmatrix}$
3. $(\beta_{(1)} A')^{-1}$
4. $(\beta_{(1)} A')^{-1} V'_{(1)} = W_{(1)}$
5. $V'_{(2)} - \beta_{(2)} A' W_{(1)} = W_{(2)}$.

Of course, actual computations in steps (1) and (2), are taken care by the LU decomposition of B, rather than by the inverse form β, which is used only for ease in symbolism.

If V^T is a row-vector, the computation of $W^T = V^T B'^{-1} = V^T E B^{-1}$ is as simple: compute $V^T E$, then $(V^T E)\, B^{-1}$.

Returning now to the exemple, we have:

$$B^{-1} = \frac{1}{4} \begin{array}{|c|c|c|} \hline 3 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 3 \\ \hline \end{array}$$

$$A = \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix} \qquad A' = \begin{bmatrix} -1 \\ 1/2 \\ -1 \end{bmatrix} \text{ (second column)}$$

$$\beta_{(1)} = \frac{1}{4} (2, 4, 2) \qquad \text{(second row)}$$

$$\beta A' = \begin{pmatrix} -3/4 \\ -1/2 \\ -3/4 \end{pmatrix}, \quad \beta_{(1)} A' = -1/2, \quad (\beta_{(1)} A')^{-1} = -2.$$

One can check that:

$$B'^{-1} = -\frac{1}{2} \begin{array}{|c|c|c|} \hline 0 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 0 \\ \hline \end{array} = EB^{-1}$$

$$= \begin{array}{|c|c|c|} \hline 1 & -3/2 & \\ \hline & -2 & \\ \hline & -3/2 & 1 \\ \hline \end{array} \times \frac{1}{4} \begin{array}{|c|c|c|} \hline 3 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 3 \\ \hline \end{array}$$

As still another instance of the usage of two bases, suppose the jacobian is:

	1	2	3	4
I_1	2	−1		
I_2	−1	2	−1	−1/2
I_3		−1	2	−1

again with three nodes I_1, I_2, I_3, and $P(I_1) = \{1\}$, $P(I_2) = \{2\}$, $P(I_3) = \{3, 4\}$. Assume furthermore that $0 \leqslant x_j \leqslant 1$, $j = 1, \ldots, 4$, defines the parallelotope, and that $x_1^0 = 1/2$, $x_2^0 = 1$, $x_3^0 = x_4^0 = 1/2$. The basis for the GRG method is:

$$B' = \begin{array}{|c|c|c|} \hline 2 & & \\ \hline -1 & -1/2 & -1 \\ \hline & -1 & 2 \\ \hline \end{array}$$

while we choose B as

$$B = \begin{array}{|c|c|c|} \hline 2 & -1 & \\ \hline -1 & 2 & -1 \\ \hline & -1 & 2 \\ \hline \end{array}$$

Here, we have

$$A = \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}, \quad A' = \begin{bmatrix} 0 \\ -1/2 \\ -1 \end{bmatrix} \quad \text{(second column)}$$

$$\beta A' = \begin{bmatrix} -1/2 \\ -1 \\ -1 \end{bmatrix}, \quad \beta_{(1)} A' = -1, \quad (\beta_{(1)} A')^{-1} = -1.$$

One can check that $B'^{-1} = E\, B^{-1}$ gives:

$$B'^{-1} = \begin{array}{|c|c|c|} \hline 1 & -1/2 & \\ \hline & -1 & \\ \hline & -1 & 1 \\ \hline \end{array} \times \frac{1}{4} \begin{array}{|c|c|c|} \hline 3 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 3 \\ \hline \end{array} =$$

$$= \begin{array}{|c|c|c|} \hline 1/2 & 0 & 0 \\ \hline -1/2 & -1 & -1/2 \\ \hline -1/4 & -1/2 & 1/4 \\ \hline \end{array}.$$

The efficiency in using two bases stems from the fact that the number of columns which are common to B and B' should be large, the number of columns in A and A' being small.

For an early use of two bases, in a less general framework, see G. DANTZIG [9].

The LU decomposition has been first proposed, for general and some special linear programming problems, by G. DANTZIG [8].

REFERENCES

1. ABADIE, J.: *Application of the GRG Algorithm to Optimal Control Problems.* Ch. 8 of: Integer and Nonlinear Programming North-Holland Publishing Company, Amsterdam and American Elsevier, New York, 1970.
2. ABADIE, J. and CARPENTIER, J.: *Généralisation de la méthode du gradient réduit de Wolfe au cas des contraintes non-linéaires.* EDF Note HR 6678, 1965.
3. ABADIE, J. and CARPENTIER J.: *Généralisation de la méthode du gradient réduit de Wolfe au cas des contraintes non-linéaires.* Proceedings of the Fourth IFORS Conference, D.B.Hertz and J. Melese, ed. Wiley Interscience, 1963.
4. ABADIE, J. and CARPENTIER Y.: *Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear constraints.* In: Optimization R. Fletcher, ed., Academic Press, 1969.
5. ABADIE, J. and GUIGOU J.: *Numerical Experiments with the GRG method.* Appendix III in: Integer and Nonlinear Programming, North-Holland Publishing Company, Amsterdam, and American Elsevier, New York, 1970.
6. BARTELS, R. H. and GOLUB G.H.: *The Simplex Method of Linear Programming using LU Decomposition.* Comm. ACM, *12* (1969), pp. 266—268.
7. COLVILLE, A.R.: *A Comparative Study of Nonlinear Programming Code IBM.* NYSC Report 320—2949 (available on writing to NYSC, IBM, 410 East 62nd St.,
8. DANTZIG, G.B.: *Upper Bounds, Secondary Constraints and Block Triangularity in Linear Programming.* Econometrica, *23* (1955), pp. 174—183.
9. DANTZIG, G.B.: *Compact Basis Triangularization for the Simplex Method.* In: Recent Advances in Mathematical Programming, R.L. Graves and P. Wolfe (eds.), McGraw-Hill, 1963.
10. DOMMEL, H.W. and TINNEY W.F.: *Optimal Power Flow Solutions.* IEEE Transactions on Power Apparatus and Systems, October 1968, pp. 1866—1875.
11. ROSEN J.B.: *The Gradient Projection Method for Nonlinear Programming.* Part II, Nonlinear Constraints, SIAM J., *9* (1961), pp. 514—532.
12. WOLFE, P.: *Methods for linear contraints,* Ch. 6 of: Nonlinear Programming, North-Holland Publishing Company Amsterdam, 1967.
13. ZOUTENDIJK, G.: *Computational Methods.* Ch. 2 of: Integer and Nonlinear Programming, North-Holland Publishing Company, Amsterdam and American Elsevier, New York, 1970.

ЗАДАЧИ ОПТИМИЗАЦИИ С СОЕДИНЕННЫМИ БЛОКАМИ

Решение задач оптимизации больших размеров находится еще в стадии экспериментирования при помощи различных методов, предлагаемых разными авторами. В настоящей статье описывается в качестве подкласса метода касательных направлений метод обобщенного сокращенного градиента, предлагаемого Ж. Карпантье и автором и сокращенно именуемого GRG. Описание сопровождается решением отдельных практических задач при помощи предлагаемого метода.

PROBLEMES D'OPTIMISATION A BLOCS COUPLES

La résolution des problèmes d'optimisation ayant de grandes dimensions se trouve encore en phase d'expérimentation au moyen de différentes méthodes appartenant à divers auteurs. La présente étude porte sur la méthode du gradient réduit généralisé, en tant que sous-classe de la méthode des directions tangentes. La méthode du gradient réduit généralisé est une méthode appartenant à J. Carpentier et à l'auteur et est nommée tout court GRG. L'exposition est accompagnée par la résolution, à l'aide de cette méthode, de quelques problèmes pratiques.