REVUE FRANÇAISE D'AUTOMATIQUE, D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE. RECHERCHE OPÉRATIONNELLE

E. CASAS
C. POLA

An algorithm for indefinite quadratic programming based on a partial Cholesky factorization

Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle, tome 27, n° 4 (1993), p. 401-426.

<http://www.numdam.org/item?id=RO_1993__27_4_401_0>

Recherche opérationnelle/Operations Research
(vol. 27, n° 4, 1993, p. 401 à 426)

AN ALGORITHM FOR INDEFINITE QUADRATIC PROGRAMMING BASED ON A PARTIAL CHOLESKY FACTORIZATION (*)

by E. CASAS ([1]) and C. POLA ([2])

Abstract. – *A new algorithm is described for quadratic programming that is based on a partial Cholesky factorization that uses a diagonal pivoting strategy and allows computation of null of negative curvature directions. The algorithm is numerically stable and has shown efficiency solving positive-definite and indefinite problems. It is specially interesting in indefinite cases because the initial point does not need to be a vertex of the feasible set. We thus avoid introducing artificial constraints in the problem, which turns out to be very efficient in parametric programming. At the same time, techniques for updating matrix factorizations are used.*

Keywords: Quadratic Programming, Cholesky Factorization, Negative, Null and Positive Curvature Directions.

Résumé. – *Nous présentons dans cet article un nouvel algorithme de programmation quadratique qui repose sur une factorisation de Cholesky avec une stratégie de pivotation diagonale et qui permet de calculer des directions de curvature nulle ou négative. L'algorithme est numériquement stable et il a montré son efficacité pour résoudre des problèmes définis positifs et indéfinis. Il est notamment intéressant dans les cas indéfinis parce que le point initial n'a pas besoin d'être un extrême de la région de points admissibles. Donc nous évitons d'introduire des contraintes artificielles dans le problème, ce qui s'avère très efficace dans la programmation paramétrique. En même temps nous utilisons des techniques pour adapter les factorisations des matrices.*

Mots clés : Programmation quadratique ; factorisation de Cholesky ; directions de courbure nulle, positive ou négative.

1. INTRODUCTION

The aim of this paper is to present a new algorithm for solving the following quadratic problem:

(*) This research was partially supported by Dirección General Científica y Técnica (Madrid).
Received May 1992.

([1]) Departamento de Matemática Aplicada y Ciencias de la Computación, Universidad de Cantabria, 39071 Santander, Spain.

([2]) Departamento de Matemáticas, Estadística y Computación, Universidad de Cantabria, 39071 Santander, Spain.

Recherche opérationnelle/Operations Research, 0399-0559/93/04/$ 4.00

$$\left.\begin{array}{rl} \text{Minimize} & F(x) = \dfrac{1}{2} x^T H x + p^T x \\ \text{subject to} & c_j^T x = b_j, \qquad 1 \leq j \leq m_e, \\ & c_j^T x \leq b_j, \qquad m_e + 1 \leq j \leq m_e + m_i, \\ & l_i \leq x_i \leq u_i, \qquad 1 \leq i \leq n, \end{array}\right\} \tag{QP}$$

where H is an $n \times n$ symmetric matrix, p and c_j are n-vectors, b_j are real numbers and l_i and u_i are elements of $[-\infty, +\infty]$ satisfying: $l_i \leq u_i$. In the sequel we will denote by C the $n \times m$ matrix that collects the column vectors c_j and b will be the vector $\{b_j\}_{j=1}^{m}$, with $m = m_e + m_i$.

Although a quadratic programming code must consider bound constraints separately from more general inequality constraints $c_j^T x \leq b_j$, in order to simplify the exposition were are going to formulate the problem as

$$\left.\begin{array}{rl} \text{Minimize} & F(x) = \dfrac{1}{2} x^T H x + p^T x \\ \text{subject to} & c_j^T x = b_j, \qquad 1 \leq j \leq m_e, \\ & c_j^T x \leq b_j, \qquad m_e + 1 \leq j \leq m_e + m_i. \end{array}\right\} \tag{QP}$$

There are two kinds of active-set methods to solve this problem. The first kind follows a strategy for choosing a certain subset of active constraints (the working set) that ensures that the reduced Hessian respect to the working set never has more than one nonpositive eigenvalue. Almost all these methods start from a feasible point x^0 that is a vertex of the feasible region, or else it is necessary to add artificial constraints to the problem so that the initial point is a vertex. These constraints are deleted from the working set as soon as possible, which means that the algorithm must perform at least as many iterations as the number of artificial constraints that have been added; *see* Fletcher ([3, 4]), Gill and Murray [7] and Gill *et al.* [9]. This can retard the solution of the problem, particularly in parametric programming when the active constraints are close to being identified. Nevertheless there is a method belonging to this king, proposed recently by Gill *et al.* [8], that does not need to start from a vertex of the feasible region. The idea is to add only a minimum number of artificial constraints to cover any non-positive curvature in the reduced Hessian. However it is not possible to know *a priori* this minimum number and sometimes a great deal of unnecessary artificial constraints are added; *see* Example 7, in Section 5.

The second kind of methods allows any number of nonpositive eigenvalues in the reduced Hessian and therefore it does not need to start from a vertex of the feasible region. Our method belongs to this kind. An advantage of our

strategy is that the algorithm has superior theoretical convergence properties than the algorithms allowing only a nonpositive eigenvalue. Indeed, if the algorithm computes a Kuhn-Tucker point at which some of the Lagrange multipliers corresponding to active constraints are null and the corresponding reduced Hessian is singular and positive semi-definite, our method allows to remove the inequality constraints associated with a null Lagrange multiplier and to proceed towards the solution if the new reduced Hessian has a negative curvature direction and if degeneracy does not occur. If the new reduced Hessian continues to be positive semi-definite, then we can repeat the process removing another inequality constraint with a null Lagrange multiplier. In constrat, the strategy mentioned above does not allow to leave the Kuhn-Tucker point; *see* for instance Gill *et al.* [8]. Nevertheless, there is a suggested procedure to deal with these situations given by Forsgren *et al.* [5].

In general, the algorithms allowing only a nonpositive eigenvalue can not solve a quadratic programming problem when the reduced Hessian at the solution is singular and the eigenvalue zero has multiplicity two or more. However, our algorithm follows a strategy that allows to solve these problems. In Section 5, we will present two examples of this kind of problems.

Another method allowing any number of nonpositive eigenvalues in the reduced Hessian is due to Bunch and Kaufman [1]. Their method is based on the decomposition $Q=MDM^T$ of a symmetric matrix Q, where D is block diagonal with blocks of order 1 or 2, and M is the product of permutations and block elementary transformations. Our algorithm is based on a partial Cholesky factorization.

In this paper we will see that it is possible to get feasible descent directions of negative, null or positive curvature from the Cholesky decomposition of the reduced Hessian, thereby allowing us to deal with any case of indefinite quadratic programming. This is performed without introducing any artificial constraints. Also, the algorithm may be started at any feasible point x^0. Obviously numerical stability requires that the factorization be stopped when indefiniteness of matrix is detected. In this case we compute a negative curvature direction from the partial factorization. When the matrix is positive semi-definite it is possible to carry out the complete factorization and to derive a null or positive curvature descent direction. We also show that it is possible to update the Cholesky factors in all cases in a similar form to that used by Gill and Murray [7].

The plan of this paper is the following. In the next section we study the Cholesky factorization of a symmetric matrix A and show the way of getting negative or null curvature directions. In Section 3 the proposed quadratic programming algorithm is stated. Updating of matrix factors is considered in Section 4 and some numerical examples are studied in Section 5.

2. SOME QUESTIONS ABOUT CHOLESKY DECOMPOSITION

Given an $n \times n$ matrix A that is symmetric but not necessarily positive definite, we are going to propose an algorithm that supplies the Cholesky decomposition of PAP^T (where P is a permutation matrix) and a basis of its kernel when A is positive semi-definite and that realizes an incomplete factorization and furnishes a negative curvature direction when A is indefinite. We first establish the following theorem.

THEOREM 1: *A matrix A is positive semi-definite if and only if there exists a permutation matrix P such that*

$$PAP^T = \begin{pmatrix} L & 0 \\ B & 0 \end{pmatrix} \begin{pmatrix} L^T & B^T \\ 0 & 0 \end{pmatrix},$$

where L is an $m \times m$ lower-triangular matric with strictly positive diagonal elements and B is an $(n-m) \times m$ matrix. The rank of A is m and a basis of the kernel of A is formed by the vectors $\{P^T u_j\}_{j=1}^{n-m}$ defined by

$$u_j = \begin{pmatrix} \hat{u}_j \\ 0 \end{pmatrix} - e_{m+j},$$

where e_{m+j} is the $(m+j)-th$ column of the $n \times n$ identity matrix and $\hat{u}_j$ is the m-vector solution of the system $L^T \hat{u}_j = B_j^T$, with B_j being the j-th row of B.

Proof: It is easy to verify that $\{P^T u_j\}_{j=1}^{n-m}$ is a basis of the kernel of A. On the other hand it is well known that A is positive semi-definite if and only if the above factorization is possible, *see* for example [2].

Now we propose an algorithm that determines if A is positive semi-definite or indefinite and performs the Cholesky decomposition in the first case.

ALGORITHM: 1. Set $k=1$, $A^{(k)}=\left(a_{ij}^{(k)}\right)=A$, $P^{(k)}=$Identity and

$$\beta = 1.2 \cdot (\max \{|a_{jj}|, j = 1, ..., n\})^{1/2}.$$

2. Find q such that

$$a_{qq}^{(k)} = \max \{a_{jj}^{(k)}, j = k, ..., n\}.$$

If $a_{qq}^{(k)} < 0 \Rightarrow$ Indefinite matrix. STOP.

If $a_{qq}^{(k)} = 0 \Rightarrow$ Find t such that

$$a_{tt}^{(k)} = \min \{a_{jj}^{(k)},\ j = k, ..., n\}.$$

• If $a_{tt}^{(k)} < 0 \Rightarrow$ Indefinite matrix. STOP.

• If $a_{tt}^{(k)} = 0 \Rightarrow$ Find r and s such that

$$|a_{rs}^{(k)}| = \max \{|a_{ij}^{(k)}|,\ n \geq i > j \geq k\}.$$

–If $a_{rs}^{(k)} \neq 0 \Rightarrow$ Indefinite matrix. STOP.

–If $a_{rs}^{(k)} = 0 \Rightarrow$ End of factorization. STOP.

If $a_{qq}^{(k)} > 0 \Rightarrow$ Interchange the rows and columns q and k of $A^{(k)}$, the new matrix being denoted again by $A^{(k)}$. Perform the same interchange of rows in $P^{(k)}$ and denote the new matrix by $P^{(k+1)}$.

3. Apply the following formulas:

For $j=1$ to $k-1$

$$a_{ij}^{(k+1)} = a_{ij}^{(k)}, \qquad i = j, ..., n, \qquad a_{kk}^{(k+1)} = \sqrt{a_{kk}^{(k)}}.$$

If $k = n \Rightarrow$ End of factorization. STOP.

For $i = k + 1$ to n

$$a_{ik}^{(k+1)} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k+1)}}.$$

If $\max \{|a_{ik}^{(k+1)}|,\ i = k + 1, ..., n\} > \beta \Rightarrow$ Indefinite matrix. STOP.

For $j = k + 1$ to n

$$a_{jj}^{(k+1)} = a_{jj}^{(k)} - \left(a_{jk}^{(k+1)}\right)^2,$$

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - a_{ik}^{(k+1)}\, a_{jk}^{(k+1)}, \qquad i = j + 1, ..., n.$$

4. Set $k=k+1$. Go to Step 2.

Remarks: (1) In practice the determination of A as a positive semi-definite matrix is based on the choice of a parameter *TOL*, which depends of the size of A and machine precision ϵ_M. We have taken

$$TOL = n \cdot \max \{1,\ |a_{jj}|,\ j = 1, ..., n\} \cdot \epsilon_M.$$

We thus decide than an element a_{ij} is zero if $|a_{ij}| < TOL$.

(2) Note that interchanges permit the factorization to progress until all remaining diagonal elements are null or negative. This will be useful for our quadratic programming algorithm, but in order to preserve numerical stability it is necessary to control the growth of the Cholesky factors. We have therefore incorporated a parameter β into our decomposition. From Cholesky formulas it follows that if

$$|a_{jk}^{(k+1)}|^2 > a_{jj}^{(k)} \quad \Rightarrow \quad a_{jj}^{(k+1)} < 0$$

and then A is an indefinite matrix. Also, from the Cholesky formulas we get that $a_{jj}^{(k)}$ is smaller than the corresponding initial diagonal element of A, thus we have

$$a_{jj}^{(k)} \leq \max \{a_{ii},\ i = 1, ..., n\}.$$

Therefore, if $|a_{jk}^{(k+1)}| > \beta$, we deduce from the previous relations that A is an indefinite matrix. No special meaning must be attributed to the factor 1.2 in the definition of β, any number greater than one would be correct.

(3) If the algorithm is stopped at iteration k with an indication of indefiniteness, then we have obtained a permutation matrix $P = P^{(k)}$ and the following factorization:

$$PAP^T = \begin{pmatrix} L & 0 \\ B & I_{n-m} \end{pmatrix} \begin{pmatrix} I_m & 0 \\ 0 & D \end{pmatrix} \begin{pmatrix} L^T & B^T \\ 0 & I_{n-m} \end{pmatrix},$$

where $m = k-1$, L is an $m \times m$ nonsingular lower-triangular matrix, B is $(n-m) \times m$ and D is an $(n-m) \times (n-m)$ symmetric matrix having (at least) one strictly negative curvature direction. These matrices are defined by the equalities

$$\begin{aligned} l_{ij} &= a_{ij}^{(k)}, && 1 \leq j \leq i \leq m, \\ d_{ij} &= a_{i+m,\, j+m}^{(k)}, && 1 \leq j \leq i \leq n-m, \\ b_{ij} &= a_{m+i,\, j}^{(k)}, && 1 \leq i \leq n-m, \quad 1 \leq j \leq m. \end{aligned}$$

From this factorization we can obtain some negative curvature directions. First let us suppose that $a_{jj}^{(k)} < 0$ for some index $j \geq k$; then the n-vector $P^T u$, with u defined by the equality

$$u = \begin{pmatrix} \hat{u} \\ 0 \end{pmatrix} + e_j,$$

where $L^T\hat{u} = -B_{j-m}^T$ and B_{j-m} is the $(j-m)$-th row of B, is a negative curvature direction of A:

$$
\begin{aligned}
&(P^T u)^T A (Pu) \\
&= \left(\begin{pmatrix}\hat{u}\\0\end{pmatrix} + e_j\right)^T \begin{pmatrix}L & 0\\ B & I_{n-m}\end{pmatrix}\begin{pmatrix}I_m & 0\\ 0 & D\end{pmatrix}\begin{pmatrix}L^T & B^T\\ 0 & I_{n-m}\end{pmatrix}\left(\begin{pmatrix}\hat{u}\\0\end{pmatrix} + e_j\right) \\
&= \left(\begin{pmatrix}L^T\hat{u} + B_{j-m}^T\\ 0\end{pmatrix} + e_j\right)^T \begin{pmatrix}I_m & 0\\ 0 & D\end{pmatrix}\left(\begin{pmatrix}L^T\hat{u} + B_{j-m}^T\\ 0\end{pmatrix} + e_j\right) \\
&= e_j^T \begin{pmatrix}I_m & 0\\ 0 & D\end{pmatrix} e_j = a_{jj}^{(k)} < 0.
\end{aligned}
$$

Now suppose that $a_{jj}^{(k)} = 0$ for $j=k$ to n and $|a_{rs}^{(k)}| > 0$ for some r, s, with $k \le s < r \le n$. In this case we take the n-vector

$$u = \begin{pmatrix}\hat{u}\\0\end{pmatrix} - a_{rs}^{(k)} e_s + e_r,$$

where $L^T\hat{u} = -B_{r-m}^T + a_{rs}^{(k)} B_{s-m}^T$, and we see that $P^T u$ is a negative curvature vector of A:

$$
\begin{aligned}
(P^T u)^T A (Pu) &= u^T \begin{pmatrix}L & 0\\ B & I_{n-m}\end{pmatrix}\begin{pmatrix}I_m & 0\\ 0 & D\end{pmatrix}\begin{pmatrix}L^T & B^T\\ 0 & I_{n-m}\end{pmatrix} u \\
&= \left(-a_{rs}^{(k)} e_s + e_r\right)^T \begin{pmatrix}I_m & 0\\ 0 & D\end{pmatrix}\left(-a_{rs}^{(k)} e_s + e_r\right) \\
&= \left(a_{rs}^{(k)}\right)^2 d_{s-m,\,s-m} + d_{r-m,\,r-m} - 2\, a_{rs}^{(k)}\, d_{r-m,\,s-m} \\
&= \left(a_{rs}^{(k)}\right)^2 a_{ss}^{(k)} + a_{rr}^{(k)} - 2\left(a_{rs}^{(k)}\right)^2 < 0.
\end{aligned}
$$

In either case, we have seen that it is not numerically difficult to get negative curvature directions of A from the partial Cholesky factorization furnished by the above algorithm. If the matrix A is positive semi-definite, Theorem 1 supplies a procedure for obtaining a basis of the kernel of A. In both cases the method is based on the solution of a system of linear equations with a matrix L^T that is upper-triangular.

3. THE MODEL ALGORITHM FOR QUADRATIC PROGRAMMING

As usual we consider an iterative procedure that follows the active-set strategy for solving quadratic programming problems. In each iteration we have a feasible point x^k, an $n \times m_k$ full column rank matrix C_k whose i-th

column contains the coefficients of the i-th active constraint in the working set at iteration k. We will denote by I_k the index set corresponding to C_k, and we will refer to I_k or C_k as the working set.

Following Gill and Murray [7], we factorize the matrix C_k into the product $Q_k R_k$, where Q_k is an orthogonal matrix and R_k is an upper-triangular matrix. In Q_k and R_k we distinguish the submatrices:

$$Q_k = (Y_k \; S_k) \qquad \text{and} \qquad R_k = \begin{pmatrix} \hat{R}_k \\ 0 \end{pmatrix},$$

where Y_k is $n \times m_k$, S_k is $n \times (n-m_k)$ and $\hat{R}_k$ is $m_k \times m_k$. The columns of Y_k form an orthonormal basis for the range space of C_k and those of S_k form an orthonormal basis for the null space of C_k^T. Finally we get the reduced Hessian H_k and compute its Cholesky decomposition in the way described in the previous section (note that H_k is symmetric).

The reduced Hessian could be computed by the formula $H_k = S_k^T H S_k$, but we prefer to define $H_k = Z_k^T H Z_k$ for reasons that we will explain in Section 4, where Z_k is the matrix obtained from S_k by setting its columns in reverse order, that is to say $Z_k = S_k \tilde{I}$, $\tilde{I}$ being the matrix of order $n-m_k$:

$$\tilde{I} = \begin{pmatrix} 0 & & 1 \\ & \ddots & \\ 1 & & 0 \end{pmatrix}.$$

It is well known that there are other ways to get a basis of the null space of C_k^T, mainly based on the use of LU factorization; *see* Fletcher [4]. Here we prefer the QR factorization because of its good properties of numerical stability (Wilkinson [12]) and because we are assuming that the matrices H and C are not sparse and can be stored explicity in the main memory of the computer. For large sparse matrices the LU factorization might be preferable.

Our quadratic programming algorithm performs the following steps:

Quadratic programming algorithm

1. Compute a feasible initial point x^0 and set $k=0$. Compute the factorization QR of the working set C_k, get Z_k and the reduced Hessian H_k. Apply the Cholesky decomposition algorithm to H_k as indicated above.

2. If H_k is not positive semi-definite or $Z_k^T \nabla F(x^k) \neq 0 \Rightarrow$ go to 3.
Otherwise compute the Lagrange multipliers λ^k through the equation

$$\hat{R}_k \lambda = -Y_k^T \nabla F(x^k).$$

If all Lagrange multipliers associated with the active inequality constraints are positive then we have found a local solution of the problem. STOP.

Otherwise we remove the inequality constraint corresponding to the most negative Lagrange multiplier and modify the *QR* factors of the working set and the Cholesky decomposition of the reduced Hessian. Go to Step 3.

3. Compute a descent direction:

• If H_k is positive definite, then solve the system

$$H_k d_{Z_k} = -Z_k^T \nabla F(x^k),$$

using the Cholesky factorization and take $d^k = Z_k d_{Z_k}$. Go to Step 4.

• If H_k is positive semi-definite, then compute a basis $\{u_j\}_{j=1}^{n-t_k}$ of the null space of H_k from the formulas given in Theorem 1 and take

$$\hat{d}^k = -Z_k U_k U_k^T Z_k^T \nabla F(x^k),$$

where U_k is the matrix whose columns are the vectors u_j.

– If $\hat{d}^k \neq 0$, set $d^k = \hat{d}^k$ and go to Step 5.

– If $\hat{d}^k = 0$, solve the system

$$H_k d_{Z_k} = -Z_k^T \nabla F(x^k)$$

and take $d^k = Z_k d_{Z_k}$ and go to Step 4.

• If H_k is not positive semi-definite, then compute a descent direction of negative curvature and continue at Step 5.

4. Compute

$$\rho_k = \min \left\{ 1, \min_{j \notin I_k,\, c_j^T d_k > 0} \frac{b_j - c_j^T x^k}{c_j^T d^k} \right\}.$$

Take $x^{k+1} = x^k + \rho_k d^k$ and set $k = k+1$. If $\rho_k = 1$ then go to Step 2, otherwise continue at Step 6.

5. Compute

$$\rho_k = \min_{j \notin I_k,\, c_j^T d_k > 0} \frac{b_j - c_j^T x^k}{c_j^T d^k}.$$

If the index set where we look for the minimum is empty, then the quadratic problem is unbounded below in the feasible region, so there is no finite solution. STOP.

Otherwise take $x^{k+1} = x^k + \rho_k d^k$ and set $k = k+1$. Go to Step 6.

6. If ρ_k is the step corresponding to the constraint with index j_k, add j_k to I_k and c_{j_k} to C_k. Modify the *QR* factorization and Cholesky decomposition of the new reduced Hessian. Go back to Step 2.

Remarks: (1) First we must remark that the initial point x^0 need not be a vertex. Nor do we need to introduce any artificial constraints. The Cholesky algorithm proposed in Section 2 allows us to compute a descent direction d^k in a stable way beginning with any symmetric matrix H_k. Among the possible descent directions computed by our quadratic programming algorithm, we find null, positive or negative curvature directions. In order to compute a negative curvature direction we look for the most negative element a_{jj} for the Cholesky factorization of H_k and then we take the vector $d_{Z_k} = \pm P^T u$ as indicated in Section 2. The sign is chosen in such a way that $d^k = Z_k d_{Z_k}$ is a descent direction. It H_k is indefinite, but every diagonal element a_{ii} is zero or positive, then we take a_{rs} as in the factorization algorithm and the associated negative curvature direction in the way indicated in Section 2.

(2) In practive the determination of $Z_k^T \nabla F(x^k)$ and $\hat{d}^k$ as null vectors (in steps 2 and 3 of our algorithm) is based on the comparison of their norm with a small parameter depending on machine precision.

(3) With zero Lagrange multipliers, once the reduced Hessian has ceased to be positive definite, no further constraints are deleted by an inertia-controlling algorithm, *see* Gill *et al.* [8], p. 8. However our code deletes inequality constraints with zero Lagrange multipliers, provided that cycling does not occur. In Step 2, if the smallest Lagrange multiplier associated with an active inequality constraint is zero then the corresponding constraint is removed from the working set. If the new reduced Hessian has a negative eigenvalue, then a descent direction of negative curvature is computed; else the new reduced Hessian is positive semi-definite and the following null Lagrange multiplier is studied.

Cycling can arise if degeneracy occurs at some point. Following Fletcher [4] we will say that degeneracy occurs at a point x^k if there exists a constraint j such that $j \notin I_k$, $c_j^T x^k = b_j$, and $c_j^T d^k > 0$. In this case the algorithm is forced to take a zero step and add a constraint to the working set without moving. So it is not possible to decrease F in this iteration. Although changes in the working set can be made, if degeneracy occurs at certain stationary points at which the reduced Hessian is positive semi-definite the possibility of cycling arises and this is the only way of avoiding the algorithm progress. Cycling can affect to any quadratic programming algorithm. It is easy to check, *see* for instance [11], that degeneracy can occur in that kind of points only if one of the following conditions is statisfied

- The normal vectors c_i to the active constraints are linearly dependent.

• All the Lagrange multipliers are nonnegative and at least two of them are zero.

So we can infer the following theorem.

THEOREM 2: *If the objective function is bounded below in the feasible region and degeneracy does not occur at stationary points, the previous algorithm converges in a finite number of iterations to a local minimum.*

Proof: Before removing an active constraint from the working set, the algorithm has found the minimum of the current equality constrained problem. Therefore, after a finite number of iterations the algorithm must find a point x and a set of active constraints where the reduced Hessian is positive semi-definite, the reduced gradient is null and the Lagrange multipliers corresponding to the inequality constraints of the working set are strictly positive. Under these conditions it is well known that x is a local minimum; *see* for example Fletcher [4]. ■

It is important to remark here the necessity for the Lagrange multipliers associated with inequality constraints to be strictly positive if we want to be sure that $\bar{x}$ is a local minimum. Indeed, let us consider the following example:

$$\begin{aligned} &\text{Minimize} \quad F(x) = x_3^2 - 2x_1x_2 \\ &\text{subject to} \quad 0 \leqq x_1 + x_2 \leqq 2, \\ &\qquad\qquad x_1 - x_2 \leqq -2. \end{aligned}$$

Let $\overline{x} = (-1,\ 1,\ 0)^T$ and $\overline{\lambda} = (0,\ 2)^T$. Then $(\overline{x},\ \overline{\lambda})$ is a Kuhn-Tucker point and the reduced Hessian

$$Z^T H^Z = (0,\ 0,\ 1) \begin{pmatrix} 0 & -2 & 0 \\ -2 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = 2$$

is positive definite. However $\overline{x}$ is not a local minimum because $F(x_\epsilon) < F(\overline{x})$ for each $\epsilon \neq 0$, where $x_\epsilon = (\epsilon - 1, \epsilon + 1, 0)^T : F(x_\epsilon) = 2(1-\epsilon^2) < 2 = F(\overline{x})$.

(4) In Step 3 of our algorithm, there are two different ways to compute the descent direction if the reduced Hessian is positive semi-definite. First the projection of the reduced gradient on the null space of the reduced Hessian is computed. If this projection is null, it is not possible to get a descent direction of null curvature. However, the system

$$H_k d_{Z_k} = -Z_k^T \nabla F(x^k)$$

has at least one solution (in fact it has many solutions) because of the orthogonality of $Z_k^T \nabla F(x^k)$ on the null space of H_k. In this way we obtain

a descent direction of positive curvature, which allows us to solve the above example. When $Z_k^T \nabla F(x^k)$ is not orthogonal to the kernel of H_k, the above system has no solution, but fortunately we can compute a descent direction of null curvature in this case, as pointed out in the algorithm.

The solution of the above system, when it exists, is computed in the following way. First we have computed the Cholesky factorization of the reduced Hessian *(see* Theorem 1):

$$H_k = P_k^T \begin{pmatrix} L_k & 0 \\ B_k & 0 \end{pmatrix} \begin{pmatrix} L_k^T & B_k^T \\ 0 & 0 \end{pmatrix} P_k,$$

where L_k is $t_k \times t_k$ and B_k is $(n-m_k-t_k) \times t_k$. Now we denote by v^k the vector formed by the first t_k components of the vector $-P_k\, Z_k^T \nabla F(x^k)$, and compute w^k as solution of the system

$$L_k\, L_k^T\, w^k = v^k.$$

Finally we take

$$d_{Z_k} = P_k^T \begin{pmatrix} w^k \\ 0 \end{pmatrix}.$$

Let us verify that this vector is a solution of the above system:

$$\begin{aligned} H_k d_{Z_k} &= P_k^T \begin{pmatrix} L_k & 0 \\ B_k & 0 \end{pmatrix} \begin{pmatrix} L_k^T & B_k^T \\ 0 & 0 \end{pmatrix} P_k\, P_k^T \begin{pmatrix} w^k \\ 0 \end{pmatrix} \\ &= P_k^T \begin{pmatrix} L_k & 0 \\ B_k & 0 \end{pmatrix} \begin{pmatrix} L_k^T & B_k^T \\ 0 & 0 \end{pmatrix} \begin{pmatrix} w^k \\ 0 \end{pmatrix} \\ &= P_k^T \begin{pmatrix} L_k\, L_k^T\, w^k \\ B_k\, L_k^T\, w^k \end{pmatrix} = P_k^T \begin{pmatrix} v^k \\ B_k\, L_k^T\, w^k \end{pmatrix}. \end{aligned}$$

Let us denote by b^k the last $n-m_k-t_k$ components of $-P_k\, Z_k^T \nabla F(x^k)$, and for every index j $(1 \leqq j \leqq n - m_k - t_k)$ let $P_k^T\, u_j$ be the null vector of H_k defined as in Theorem 1:

$$u_j = \begin{pmatrix} \hat{u}_j \\ 0 \end{pmatrix} - e_{t_k+j},$$

with $L_k^T\, \hat{u}_j = (B_k)_j^T$.

Because $Z_k^T \nabla F(x^k)$ is orthogonal to each vector $P_k^T\, u_j$ we have

$$\begin{aligned} 0 &= (P_k^T\, u_j)^T\, (-Z_k^T \nabla F(x^k)) \\ &= -u_j^T\, P_k\, Z_k^T \nabla F(x^k) = u_j^T \begin{pmatrix} v^k \\ b^k \end{pmatrix} = \hat{u}_j^T\, v^k - b_j^k. \end{aligned}$$

Thus $\hat{u}_j^T v^k = b_j^k$ and therefore

$$(B_k L_k^T w^k)_j = (B_k)_j L_k^T w^k$$
$$= (L_k^T \hat{u}_j)^T L_k^T w^k = \hat{u}_j^T (L_k L_k^T w^k) = \hat{u}_j^T v^k = b_j^k.$$

Hence we conclude that

$$H_k d_{Z_k} = P_k^T \begin{pmatrix} v^k \\ B_k L_k^T w^k \end{pmatrix} = P_k^T \begin{pmatrix} v^k \\ b^k \end{pmatrix}$$
$$= -P_k^T P_k Z_k^T \nabla F(x^k) = -Z_k^T \nabla F(x^k).$$

4. MODIFICATION OF *QR* AND CHOLESKY FACTORS

It is well known that as changes are made to the working set, the *QR* and Cholesky factorizations can be modified rather than computed *ab initio*; *see* Gill *et al.* [6]. We must distinguish two cases as these changes are a consequence of adding or removing a constraint from the working set.

4.1. Deleting a constraint

Let C be the $n \times m$ working set, with $C = QR$, $Q = (Y\ S)$ and

$$R = \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix}.$$

Let us assume that the i-th constraint is deleted from the working set and let $\overline{C}$ be the associated matrix. Let us denote by $\overline{R}$ the matrix obtained from R removing the i-th column; then $\overline{C} = Q\overline{R}$. In order to reduce $\overline{R}$ to triangular form, we apply $m-i$ (2×2)-orthogonal transformations to the last columns of $\overline{R}$ and perform the corresponding modifications to Q. The latter do not affect the last $n-m$ columns of Q; hence the new orthogonal matrix will be $\overline{Q} = (\overline{Y}\ \overline{S})$ with S being augmented by a column, say that $\overline{S} = (s\ S)$ and hence $\overline{Z} = (Z\ z)$ where $z = s$.

If we denote by $H_Z = Z^T H Z$ the reduced Hessian before removing the i–th constraint, the new Hessian will be

$$H_{\overline{Z}} = \overline{Z}^T H \overline{Z} = \begin{pmatrix} Z^T \\ z^T \end{pmatrix} H (Z\ z) = \begin{pmatrix} Z^T H Z & Z^T H z \\ z^T H Z & z^T H z \end{pmatrix}.$$

Since we are deleting a constraint, H_Z must be positive semi-definite, so we know its Cholesky decomposition:

$$H_Z = P^T \begin{pmatrix} L & 0 \\ B & 0 \end{pmatrix} \begin{pmatrix} L^T & B^T \\ 0 & 0 \end{pmatrix} P,$$

where L is $t \times t$ and B is $(n-m-t) \times t$. From here it follows that

$$H_{\overline{Z}} = \begin{pmatrix} P & 0 \\ 0 & 1 \end{pmatrix}^T \begin{pmatrix} \begin{pmatrix} L & 0 \\ B & 0 \end{pmatrix} \begin{pmatrix} L^T & B^T \\ 0 & 0 \end{pmatrix} & PZ^T H\ z \\ z^T H Z P^T & z^T H\ z \end{pmatrix} \begin{pmatrix} P & 0 \\ 0 & 1 \end{pmatrix}.$$

Let a_1 be the vector formed by the t first components of $PZ^T Hz$ and a_2 the $n-m-t$ vector formed by the last components. We now take b_1 satisfying $Lb_1 = a_1$ and we distinguish two cases:

First Case: $z^T Hz - b_1^T b_1 > 0$.

In this case we take $\bar{b} = \sqrt{z^T Hz - b_1^T b_1}$ and $b_2 = (a_2 - Bb_1)/\bar{b}$ and then we have

$$H_{\overline{Z}} = \hat{P}^T \begin{pmatrix} L & 0 & 0 \\ B & b_2 & I_B \\ b_1^T & \bar{b} & 0 \end{pmatrix} \begin{pmatrix} I_L & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -b_2\, b_2^T \end{pmatrix} \begin{pmatrix} L^T & B^T & b_1 \\ 0 & b_2^T & \bar{b} \\ 0 & I_B & 0 \end{pmatrix} \hat{P},$$

where

$$\hat{P} = \begin{pmatrix} P & 0 \\ 0 & 1 \end{pmatrix}$$

and I_B and I_L denote the identity matrices with the same row dimension as B and L respectively. Now changing the order of the last row or column we get

$$H_{\overline{Z}} = \overline{P}^T \begin{pmatrix} L & 0 & 0 \\ b_1^T & \bar{b} & 0 \\ B & b_2 & I_B \end{pmatrix} \begin{pmatrix} I_L & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -b_2\, b_2^T \end{pmatrix} \begin{pmatrix} L^T & b_1 & B^T \\ 0 & \bar{b} & b_2^T \\ 0 & 0 & I_B \end{pmatrix} \overline{P}$$

$$= \overline{P}^T \begin{pmatrix} \overline{L} & 0 \\ \overline{B} & I_B \end{pmatrix} \begin{pmatrix} I_{\overline{L}} & 0 \\ 0 & -b_2\, b_2^T \end{pmatrix} \begin{pmatrix} \overline{L}^T & \overline{B}^T \\ 0 & I_B \end{pmatrix} \overline{P},$$

where $\overline{B} = (B\ b_2)$ and $\overline{P}$ is the matrix obtained from $\hat{P}$ by interchanging the appropriate rows.

If $b_2 \neq 0$ then $H_{\overline{Z}}$ has a negative curvature direction that can be computed as indicated in Section 2 from the above decomposition. When $b_2 = 0$, $H_{\overline{Z}}$ is positive semi-definite and its Cholesky decomposition is

$$H_{\overline{Z}} = \overline{P}^T \begin{pmatrix} \overline{L} & 0 \\ \overline{B} & 0 \end{pmatrix} \begin{pmatrix} \overline{L}^T & \overline{B}^T \\ 0 & 0 \end{pmatrix} \overline{P}.$$

Second Case: $z^T Hz - b_1^T b_1 \leqq 0$.

In this case we take $\bar{b} = z^T H z - b_1^T b_1$ and $u = a_2 - Bb_1$ and obtain

$$H_{\overline{Z}} = \overline{P}^T \begin{pmatrix} L & 0 & 0 \\ B & I_B & 0 \\ b_1^T & 0 & 1 \end{pmatrix} \begin{pmatrix} I_L & 0 & 0 \\ 0 & 0 & u \\ 0 & u^T & \bar{b} \end{pmatrix} \begin{pmatrix} L^T & B^T & b_1 \\ 0 & I_B & 0 \\ 0 & 0 & 1 \end{pmatrix} \overline{P}$$

$$= \overline{P}^T \begin{pmatrix} \overline{L} & 0 \\ \overline{B} & I_{\overline{B}} \end{pmatrix} \begin{pmatrix} I_L & 0 & 0 \\ 0 & 0 & u \\ 0 & u^T & \bar{b} \end{pmatrix} \begin{pmatrix} \overline{L}^T & \overline{B}^T \\ 0 & I_{\overline{B}} \end{pmatrix} \overline{P},$$

where $\overline{L} = L$, $\overline{B}^T = (B^T \; b_1)$ and

$$\overline{P} = \begin{pmatrix} P & 0 \\ 0 & 1 \end{pmatrix}.$$

The matrix $H_{\overline{Z}}$ has a negative curvature direction if $\bar{b} \neq 0$ or $u \neq 0$; otherwise it would be positive semi-definite with a Cholesky decomposition associated with the matrices $\overline{L}$ and $\overline{B}$.

If $u \neq 0$ or $\bar{b} \neq 0$ we can get a negative curvature direction for $H_{\overline{Z}}$ in the following way:

$$L^T \hat{u} = -b_1 + B^T u \qquad \text{and} \qquad d_{\overline{Z}} = \begin{pmatrix} \hat{u} \\ -u \\ 1 \end{pmatrix}.$$

It is easy to verify that $\left(\overline{P}^T d_{\overline{Z}}\right)^T H_{\overline{Z}} \overline{P}^T d_{\overline{Z}} = -2u^T u + \bar{b} < 0$.

4.2. Adding a constraint

Let C, $Q=(Y\,S)$, R and Z be as in the previous section and assume that the constraint c is to be added to the working set. we define the new working set as $\overline{C}=(C\;c)$ and we have the equality $\overline{C} = Q\,(R\;Q^T c)$. To derive the QR factorization of $\overline{C}$ we choose an orthogonal transformation to annihilate the last $n-m-1$ components of $Q^T c$. This transformation can be a Householder matrix or a product of plane rotations. Since we have to compute the new Cholesky factors of H_Z, it is preferable to apply rotations in a certain order, as we shall see. For the moment let M be a product of $n-m-1$ rotations that turn $(R\;Q^T c)$ into an upper-triangular matrix. M affects only the last $n-m$ rows of $(R\;Q^T c)$ and its form is

$$M = \begin{pmatrix} I_m & 0 \\ 0 & \hat{M} \end{pmatrix},$$

with $\hat{M}$ orthogonal (a product of plane rotations), so $\overline{Q}=QM^T=(Y\ S\hat{M}^T)$ and

$$\overline{R}=\begin{pmatrix}\hat{R} & u\\ 0 & \hat{M}v\end{pmatrix},$$

where u represents the first m components of Q^Tc and v the remainder, $\hat{M}v$ being zero except for its first component.

Now take $\hat{Z}=S\hat{M}^T\tilde{I}=(\overline{Z}\ z)$. It is clear that $\overline{Z}$ is a matrix whose columns are orthogonal and form a basis of the null space of $\overline{C}^T$. To obtain the Cholesky factors of $H_{\overline{Z}}=\overline{Z}^T H\overline{Z}$, we use the Cholesky decomposition of H_Z (*see* Section 2):

$$H_Z=P^T\begin{pmatrix}L & 0\\ B & I_B\end{pmatrix}\begin{pmatrix}I_L & 0\\ 0 & D\end{pmatrix}\begin{pmatrix}L^T & B^T\\ 0 & I_B\end{pmatrix}P,$$

where D is the null matrix if H_Z is positive semi-definite and D has at least one negative curvature direction if H_Z is indefinite. If H_Z is positive definite, the previous factorization reduces to $H_Z=P^T LL^T P$.

Let us define $\hat{H}_Z=\hat{Z}^T H\hat{Z}$. Then, from the relation $Z=S\tilde{I}$, it follows that

$$\hat{H}_Z=\hat{Z}^T H\hat{Z}=\tilde{I}\hat{M}S^T HS\hat{M}^T\tilde{I}$$

$$=\tilde{I}\hat{M}\tilde{I}Z^T HZ\tilde{I}\hat{M}^T\tilde{I}=\tilde{I}\hat{M}\tilde{I}H_Z\tilde{I}\hat{M}^T\tilde{I}$$

$$=\tilde{I}\hat{M}\tilde{I}P^T\begin{pmatrix}L & 0\\ B & I_B\end{pmatrix}\begin{pmatrix}I_L & 0\\ 0 & D\end{pmatrix}\begin{pmatrix}L^T & B^T\\ 0 & I_B\end{pmatrix}P\tilde{I}\hat{M}^T\tilde{I}.$$

If $P=I$ and the rotations are applied in the planes $(n,\ n-1)$, $(n-1,\ n-2)$, ..., $(m+2,m+1)$, then the matrix

$$X=\tilde{I}\hat{M}\tilde{I}P^T\begin{pmatrix}L & 0\\ B & I_B\end{pmatrix}$$

is lower-Hessenber of the form (*see* Gill and Murray [7])

$$X=\begin{pmatrix}r & N\\ \sigma & s^T\end{pmatrix}\quad\text{with}\quad N=\begin{pmatrix}N_{11} & 0\\ N_{21} & N_{22}\end{pmatrix}\quad\text{and}\quad r=\begin{pmatrix}r_1\\ r_2\end{pmatrix},$$

where the dimensions of N_{22} and D are equal and those of N_{11} are one less than those of L.

When P is a permutation matrix different from the identity we can obtain the same result by performing the rotations in a different order, namely $\hat{M}=\hat{M}_{i_k,j_k}\dots\hat{M}_{i_1,j_1}$, where $k=\hat{n}-1$, $\hat{n}=n-m$ being the dimension of $\hat{M}$, and $\hat{M}_{i_q,j_q}$ is a rotation in the plane $(\hat{n}+1-i_q,\ \hat{n}+1-j_q)$, $q=1,\dots,k$. The

pairs $(\hat{n}+1-i_1, \hat{n}+1-j_1), \ldots, (\hat{n}+1-i_k, \hat{n}+1-j_k)$ are formed from the vector *JPVT* that indicates the interchanges needed for the Cholesky factorization. P is a permutation matrix obtained by permuting the rows of the identity matrix, and $JPVT(q)$, $q=1, \ldots, \tilde{n}$, contains the index of the row of the identity that was moved into the q–th position. We now form the pairs (i_q, j_q) in the following way:

- If $JPVT\,(1) > JPVT\,(2) \Rightarrow j_1 = JPVT\,(1)$ and $i_1 = JPVT\,(2)$.
- Else $j_1 = JPVT\,(2)$ and $i_1 = JPVT\,(1)$
- For $q = 2$ to $\hat{n} - 1$.
  - If $j_{q-1} > JPVT\,(q+1) \Rightarrow j_q = j_{q-1}$ and $i_q = JPVT\,(q+1)$.
  - Else $j_q = JPVT\,(q+1)$ and $i_q = j_{q-1}$.
- End.

Finally we can choose a permutation matrix P_X, having the structure

$$P_X = \begin{pmatrix} \overline{P}_X & 0 \\ 0 & 1 \end{pmatrix},$$

and such that

$$X = P_X \tilde{I} \hat{M} \tilde{I} P^T \begin{pmatrix} L & 0 \\ B & I_B \end{pmatrix}$$

is lower-Hessenberg. Thus we have

$$\begin{aligned} \hat{H}_Z &= P_X^T X \begin{pmatrix} I_L & 0 \\ 0 & D \end{pmatrix} X^T P_X \\ &= P_X^T \begin{pmatrix} rr^T + N\hat{D}N^T & \sigma r + N\hat{D}s \\ \sigma r^T + s^T \hat{D} N^T & \sigma^2 + s^T \hat{D} s \end{pmatrix} P_X, \end{aligned}$$

where $\hat{D}$ is the matrix that satisfies

$$\begin{pmatrix} I_L & 0 \\ 0 & D \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & \hat{D} \end{pmatrix}.$$

On the other hand,

$$\begin{aligned} \hat{H}_Z &= \hat{Z}^T H \hat{Z} = (\overline{Z}\, z)^T H\, (\overline{Z}\, z) \\ &= \begin{pmatrix} \overline{Z}^T H \overline{Z} & \overline{Z}^T Hz \\ z^T H \overline{Z} & z^T Hz \end{pmatrix} = \begin{pmatrix} H_{\overline{Z}} & \overline{Z}^T Hz \\ z^T H \overline{Z} & z^T Hz \end{pmatrix}. \end{aligned}$$

Finally we have

$$H_{\overline{z}} = \overline{P}_X^T (rr^T + N\,\hat{D}N^T)\,\overline{P}_X$$
$$= \overline{P}_X^T \begin{pmatrix} N_{11} & 0 & r_1 \\ N_{21} & N_{22} & r_2 \end{pmatrix} \begin{pmatrix} I_{11} & 0 & 0 \\ 0 & D & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} N_{11}^T & N_{21}^T \\ 0 & N_{22}^T \\ r_1^T & r_2^T \end{pmatrix} \overline{P}_X$$
$$= \overline{P}_X^T \begin{pmatrix} N_{11} & r_1 & 0 \\ N_{21} & r_2 & N_{22} \end{pmatrix} \begin{pmatrix} I_{11} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & D \end{pmatrix} \begin{pmatrix} N_{11}^T & N_{21}^T \\ r_1^T & r_2^T \\ 0 & N_{22}^T \end{pmatrix} \overline{P}_X,$$

where I_{11} is the identity of the same order as N_{11}. It is not difficult to verify that if the i–th diagonal element of N_{11} is zero then the i-th component of r_1 is one of the diagonal elements of L and therefore it is strictly positive. Taking into acocunt this fact, it follows that it is possible to use plane rotations in order to obtain an orthogonal matrix G such that

$$G \begin{pmatrix} N_{11}^T \\ r_1^T \end{pmatrix} = \begin{pmatrix} \overline{N}_{11}^T \\ 0 \end{pmatrix},$$

$\overline{N}_{11}$ being a lower-triangular matrix with strictly positive diagonal elements. We now introduce the following notation:

$$\begin{pmatrix} \overline{N}_{11} & 0 \\ \overline{N}_{21} & \overline{r}_2 \end{pmatrix} = \begin{pmatrix} N_{11} & r_1 \\ N_{21} & r_2 \end{pmatrix} G^T$$

and

$$\overline{G} = \begin{pmatrix} G & 0 \\ 0 & I_{22} \end{pmatrix},$$

with I_{22} the identity matrix of the same order as N_{22}. From the above factorization for $H_{\overline{Z}}$ we deduce

$$H_{\overline{Z}} = \overline{P}_X^T \begin{pmatrix} \overline{N}_{11} & 0 & 0 \\ \overline{N}_{21} & \overline{r}_2 & N_{22} \end{pmatrix} \overline{G} \begin{pmatrix} I_{11} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & D \end{pmatrix} \overline{G}^T \begin{pmatrix} \overline{N}_{11}^T & \overline{N}_{21}^T \\ 0 & \overline{r}_2^T \\ 0 & N_{22}^T \end{pmatrix} \overline{P}_X$$
$$= \overline{P}_X^T \begin{pmatrix} \overline{N}_{11} & 0 \\ \overline{N}_{21} & I_{21} \end{pmatrix} \begin{pmatrix} I_{11} & 0 \\ 0 & \overline{r}_2\,\overline{r}_2^T + N_{22} D\,N_{22}^T \end{pmatrix} \begin{pmatrix} \overline{N}_{11}^T & \overline{N}_{21}^T \\ 0 & I_{21} \end{pmatrix} \overline{P}_X.$$

Finally we obtain a Cholesky factorization for $H_{\overline{Z}}$ similar to that of H_Z by performing the decomposition of the matrix $r_2\,r_2^T + N_{22}\,DN_{22}^T$.In this way we arrive at the final decomposition

$$H_{\overline{Z}} = \overline{P}^T \begin{pmatrix} \overline{L} & 0 \\ \overline{B} & I_{\overline{B}} \end{pmatrix} \begin{pmatrix} I_{\overline{L}} & 0 \\ 0 & \overline{D} \end{pmatrix} \begin{pmatrix} \overline{L}^T & \overline{B}^T \\ 0 & I_{\overline{B}} \end{pmatrix} \overline{P},$$

where the order of $\overline{D}$ is the same or lower than that of D.

In practice, in order to prevent numerical instability, $\bar{r}_2\,\bar{r}_2^T + N_{22}\,DN_{22}^T$ is not computed. The factorization of this matrix is obtained from the corresponding submatrix of $\bar{Z}^T H\bar{Z}$ following the algorithm described in Section 2.

5. NUMERICAL EXAMPLES

Our quadratic programming algorithm has been implemented in a FORTAN 77 program (OPTR04) and applied to several examples. Many of these examples were generated by using the random function of the machine, but sometimes forcing the matrix H to be singular or positive definite. Our code finished successfully in all cases, detecting an unbounded function or finding a local minimum.

Here we present seven examples. The problems were run on a VAX 8350 under VMS 5.4, in double precision (*i.e.*, machine precision is approximately 2.78×10^{-17}).

The first example, constructed by Bunch and Kaufman [1], is thoroughly documented to give the reader an idea of the course of a typical example.

EXAMPLE 1

$$\text{Minimize} \quad F(x) = \frac{1}{2}x^T Hx + p^T x,$$

where

$$h_{ij} = \begin{cases} |i-j| & \text{if} \quad i \neq j \\ 1.69 & \text{if} \quad i = j \end{cases} \qquad \text{and} \qquad p = \begin{pmatrix} 7 \\ 6 \\ \vdots \\ 0 \end{pmatrix},$$

subject to the bound constraints

$$-i - 0.1\,(i-1) \leqq x_i \leqq i, \qquad i = 1, 2, \ldots, 8$$

and the inequality constraints

$$x_i - x_{i+1} \leqq 1 + 0.05\,(i-1), \qquad i = 1, 2, \ldots, 7.$$

The problem has a local minimum of $F = -621.487825$ at the point

$$\bar{x} = (-1, -2, -3.05, -4.15, -5.3, 6, 7, 8)^T,$$

and one of -131.774167 at

$$\bar{x} = (1, 2, 1.880144, 0.780144, -0.369856, \\ -1.569856, -2.819856, -4.119856)^T.$$

The convention for numbering the constraints is the following:

$0 > i$	lower bound $-i$
$1 \leqq i \leqq n$	upper bound i
$n < i \leqq n + m_e$	equality constraint $i - n$
$n + m_e < i$	inequality constraint $i - n - m_e$.

Beginning at $x_i^0 = -i$, the routine reached the first local minimum after 7 iterations. The course of the algorithm was as follows:

Iteration 1:
- Active constraints: -1, 9.
- A descent direction of negative curvature was computed.
- Added constraint: 10.

Iteration 2:
- Active constraints: -1, 9, 10.
- A descent direction of negative curvature was computed.
- Added constraint: 11.

Iteration 3:
- Active constraints: -1, 9, 10, 11.
- A descent direction of negative curvature was computed.
- Added constraint: 12.

Iteration 4:
- Active constraints: -1, 9, 10, 11, 12.
- A descent direction of negative curvature was computed.
- Added constraint: 13.

Iteration 5:
- Active constraints: -1, 9, 10, 11, 12, 13.
- A descent direction of positive curvature was computed.
- Added constraint: 8.

Iteration 6:
- Active constraints: -1, 9, 10, 11, 12, 13, 8.
- A descent direction of positive curvature was computed.
- Added constraint: 7.

Iteration 7:
- Active constraints: -1, 9, 10, 11, 12, 13, 8, 7.
- Deleted constraint: 13.
- A descent direction of positive curvature was computed.
- Added constraint: 6.

End of Routine: A local minimum was found.

EXAMPLE 2

This example is the problem number 118 of the book of Hock and Schittkowski [10]. It is a positive definite quadratic programming exercise in 15 variables and the solution is a vertex of the feasible region and it was found by our code in 12 iterations.

EXAMPLE 3

This example is an indefinite quadratic programming problem that has been taken from the documentation of the routine E04NAF (NAG Library, Mark 14C). The feasible initial point was

$$x^0 = \begin{pmatrix} -.1000000\,E-01 \\ -.3171523\,E-01 \\ -.5983850\,E-03 \\ -.1127036\,E-01 \\ -.1000000\,E+00 \\ +.2115247\,E-01 \\ +.2431498\,E-02 \end{pmatrix},$$

and the computed solution (to seven figures)

$$\bar{x} = \begin{pmatrix} -.1000000\,E-01 \\ -.6986465\,E-01 \\ +.1825915\,E-01 \\ -.2426081\,E-01 \\ -.6200564\,E-01 \\ +.1380544\,E-01 \\ +.4066496\,E-02 \end{pmatrix}.$$

One bound constraint and four general constraints are active at the solution.

EXAMPLE 4

$$\text{Minimize} \quad F(x) = \frac{1}{2}x^T Hx + p^T x,$$

where

$$\begin{cases} H(i,\,i) = i & \text{if} \quad 1 \leqq i \leqq 10; \\ H(i,\,i+k) = 0.25ki & \text{if} \quad 1 \leqq i < 10-k, \\ H(i,i-k) = 0.25k(i-k) & \text{if} \quad k < i \leqq n, \qquad k = 1,3; \\ H(i,\,j) = 0 \quad \text{in other case;} \end{cases}$$

and

$$p(i) = (-1)^i, \qquad i = 1,\,2,\ldots,10;$$

subject to the bound constraints

$$\begin{aligned} -0.3 \leqq x_i \leqq 0.25, \qquad & i = 1, 3, ..., 9; \\ -0.3 \leqq x_i, \qquad & i = 2, 4, ..., 10; \end{aligned}$$

and the equality and inequality constraints

$$\begin{aligned} 2x_1 + 7x_2 + 3x_4 + 6x_5 + 6x_6 + 8x_7 + 6x_8 + 8x_9 &= 4, \\ 5x_1 + 6x_2 + 7x_3 + x_4 + 5x_5 + 2x_6 + 2x_7 + 2x_8 + 8x_9 + 6x_{10} &\leqq 8, \\ -3x_1 + 9x_2 - 2x_3 + 3x_4 - 3x_5 + 2x_6 - 5x_7 + 4x_8 - 3x_9 + 5x_{10} &\leqq 9, \\ 5x_1 + 4x_2 + 2x_3 + 6x_4 + 4x_5 + 9x_6 + 4x_8 + 2x_9 + 4x_{10} &\leqq 4, \\ 2x_1 - x_2 + 7x_3 - 2x_4 + x_5 - 6x_6 + x_7 - 6x_8 + 8x_9 - 4x_{10} &\leqq 5. \end{aligned}$$

Let us remark that the matrix H is positivte definite. The feasible intitial point was

$$x^0 = (.25,\ -.20174,\ .4434\,E-01,\ -.3,\ .25,\ .352,\ .25,\ -.3,\ .25,\ -.3)^T.$$

The computed solution (to seven figures) was

$$\begin{aligned} \bar{x} = (&+.1136908,\ +.2908728,\ -.3,\ -.3, \\ &+.25,\ -.1772485,\ +.25,\ -.3,\ +.25,\ -.3)^T. \end{aligned}$$

In Examples 5 and 6, the matrix H is singular, the multiplicity of zero eigen-value being two. These problems are difficult as suggested by the fact that some known routines failed to obtain a solution, as VE02AD (Harwell) and E04NAF (NAG), however both problems are bounded from below and have many infinitely solutions.

EXAMPLE 5

$$\begin{aligned} \text{Minimize} \quad & F(x) = \frac{1}{2} x^T H x + p^T x \\ \text{subject to} \quad & x_2 + 3x_3 + 2x_4 = 0, \\ & 2x_1 - x_2 + x_3 + x_4 \leqq 0, \end{aligned}$$

where

$$H = \begin{pmatrix} 4 & -2 & 2 & 2 \\ -2 & 2 & 2 & 1 \\ 2 & 2 & 10 & 7 \\ 2 & 1 & 7 & 5 \end{pmatrix} \quad \text{and} \quad p = \begin{pmatrix} 2 \\ -2 \\ -2 \\ -1 \end{pmatrix}$$

The set of solutions of this problem is the following:

$$\begin{pmatrix} -4 \\ -5 \\ 1 \\ 1 \end{pmatrix} + \alpha \begin{pmatrix} 2 \\ 3 \\ -1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 3 \\ 4 \\ 0 \\ -2 \end{pmatrix},$$

where α, β are any real numbers.

Beginning at $x^0 = (0, 0, 0, 0)^T$, our routine reached a global solution

$$\bar{x} = (-0.3414634,\ 0.2195122,\ -0.2439024\,E-01,\ -0.7317073\,E-01)^T,$$
$$F(\bar{x}) = -0.5$$

after one iteration.

The reduced Hessian at the point x^0 is the null matrix 2×2 and the reduced gradient of F at x^0 is null. Therefore an algorithm belonging to the first kind (defined in Introduction as those allowing only a nonpositive eigenvalue in the reduced Hessian) can not leave this point.

EXAMPLE 6

$$\begin{aligned} &\text{Minimize} && F(x) = \frac{1}{2} x^T H x + p^T x, \\ &\text{subject to} && -2 \leqq 0.6x_2 + 0.8x_3 \leqq 1, \\ & && x_1 - x_4 + x_5 \leqq -10, \\ & && 0 \leqq x_1 \leqq 1, \qquad -200 \leqq x_4 \leqq 5, \end{aligned}$$

where

$$H = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & 0.36 & 0.48 & 0 & 0 \\ 0 & 0.48 & 0.64 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad p = \begin{pmatrix} 2 \\ 1.2 \\ 1.6 \\ 1 \\ -7 \end{pmatrix}$$

The set of solutions of this problem is the following:

$$\{(0, \alpha, \beta, 5, -5)^T : 0.6\alpha + 0.8\beta = -2\}$$

and $F = 50.5$ for each one of these points. In this occasion, the start point was $x^0 = (0, -5, 5, 5, -5)^T$ and the solution $\bar{x} = (0, -6.8, 2.6, 5, -5)^T$.

The last example shows as algorithms allowing any number of nonpositive eigenvalues can be sometimes much faster than those allowing at most one non-positive eigenvalue, even if the method proposed by Gill *et al.* [8] is followed in order to add a minimum number of artificial constraints.

EXAMPLE 7

$$\text{Minimize} \quad F(x) = \frac{1}{2} x^T H x + p^T x,$$

where $n = 100$,

$$h_{ij} = \begin{cases} -19801 & \text{if} \quad i = j = 1 \\ -1963 & \text{if} \quad i = j > 1 \\ -11692 & \text{if} \quad j = 1 \quad \text{and} \quad 2 \leqq i \leqq 100 \\ -11692 & \text{if} \quad i = 1 \quad \text{and} \quad 2 \leqq j \leqq 100 \\ -2044 & \text{otherwise} \end{cases}, \qquad p = \begin{pmatrix} -1 \\ \vdots \\ -1 \end{pmatrix},$$

subject to the inequality constraint

$$-10 \leqq x_1 + \ldots + x_{100} \leqq +10.$$

The start point was $x^0 = 0$ and our algorithm found the solution in two iterations.

The optimum value of F was -3125243.289. The same value vas reached by E04NAF, beginning at the same point x^0, after 100 iterations. This great deal of iterations is due to the fact that E04NAF must add 100 artificial constraints (bound constraints) and then it must carry out 100 iterations to deleted them. At iteration k there are still $100-k$ null components in the vector x^k. However the matrix H has only one nonpositive eigenvalue, detected by our algorithm and corrected in the first iteration, finding the solution in the second iteration, where the reduced Hessian is already positive definite.

In Table 1 and 2, we give numerical results obtained with our code and E04NAF (NAG Library, Mark 14) respectively. The abbreviations in the tables are the following ones

TEST	Identifier for the problem.
CPU	Execution time. Number of 10 milliseconds intervals.
ITER	Number of iterations that were required to get the solution $\bar{x}$.
F	Objective function value $F(\bar{x})$.
R	Sum of constraint violations $r(\bar{x})$,

where

$$r(x) = \sum_{j=1}^{m_i} |b_j - c_j^T x| + \sum_{j=m_i+1}^{m_i+m_d} (b_j - c_j^T x)_+ + \sum_{i=1}^{n} \max(0, l_i - x_i, x_i - u_i).$$

TABLE I
Performance of OPTR04.

TEST	CPU	ITER	F	R
EX. 1 . . .	34	7	−0.6214878250 *E* + 03	+ 0.1665334537 *E* − 15
EX. 2 . . .	80	12	+ 0.6726603000 *E* + 03	+ 0.4218847494 *E* − 14
EX. 3 . . .	16	7	+ 0.3703164590 *E* − 01	+ 0.1626303259 *E* − 18
EX. 4 . . .	14	2	−0.1034296488 *E* + 02	+ 0.1110223025 *E* − 15
EX. 5 . . .	6	1	−0.5000000000 *E* + 00	+ 0.1040834086 *E* − 16
EX. 6 . . .	9	3	−0.5050000000 *E* + 02	−0.0000000000 *E* + 00
EX. 7 . . .	5522	2	−0.3125243289 *E* + 07	−0.0000000000 *E* + 00

TABLE II
Performance of E04NAF.

TEST	CPU	ITER	F	R
EX. 1 . . .	48	6	−0.6214878250 *E* + 03	+ 0.4635181128 *E* − 14
EX. 2 . . .	170	12	+ 0.6726603000 *E* + 03	+ 0.4440892099 *E* − 14
EX. 3 . . .	21	6	+ 0.3703164590 *E* − 01	+ 0.5204170428 *E* − 17
EX. 4 . . .	11	2	−0.1034296488 *E* + 02	+ 0.2220446049 *E* − 15
EX. 5 . . .	8	1	−0.0000000000 *E* + 00	+ 0.0000000000 *E* + 00
EX. 6 . . .	10	0	−0.0550000000 *E* + 02	−0.0000000000 *E* + 00
EX. 7 . . .	6759	100	−0.3125243289 *E* + 07	−0.0000000000 *E* + 00

In Examples 5 and 6, E04NAF failed to obtain a solution.

REFERENCES

1. J. R. Bunch and L. Kaufman, A Computational Method for the Indefinite Quadratic Programming Problem, *Linear Algebra and its App.*, 1980, *34*, pp. 341-370.
2. J. J. Dongarra, C. B. Moler, J. R. Bunch and G. W. Stewart, *LINPACK Users' Guide*, *SIAM*, Philadelphia, 1979.
3. R. Fletcher, A General quadratic programming, *J. Institute of Math. and its Appl.*, 1971, *7*, pp. 76-91.
4. R. Fletcher, Practical Methods of Optimization, *John Wiley and Sons*, Chichester and New York, second edition, 1987.
5. A. L. Forsgren, P. E. Gill and W. Murray, On the identificaiton of local minimizers in inertia-controlling methods for quadratic programming. *SIAM J. Matrix Anal. Appl.*, 1991, *12*, pp. 730-746.
6. P. E. Gill, G. H. Golub, W. Murray and M. A. Saunders, Methods for modifying matrix factorizations, *Mathematics of Computation*, 1974, *28*, (*126*), pp. 505-535.
7. P. E. Gill and W. Murray, Numerically stable methods for quadratic programing, *Math. Programming*, 1978, *14*, pp. 349-372.

8. P. E. GILL, W. MURRAY, M. A. SAUNDERS and M. H. WRIGHT, Inertia-controlling methods for quadratic programming, *SIAM Review*, 1991, *33*, pp. 1-36.
9. P. E. GILL, W. MURRAY and M. H. WRIGHT, Pratical Optimization, Academic Press, London and New York, 1981.
10. W. HOCK and K. SCHITTKOWSKI, Test Examples for Nonlinear Programming Codes, *Lecture in Economics and Mathematical Systems*, *Springer-Verlag*, Berlin, Heidelberg and New York, 1981.
11. C. POLA, Algorithmos Numéricos para la resolución de problemas de optimización con restricciones, *Ph. D. thesis*, Dpto. Matemáticas, Estadística y Computación, *Uniersidad de Cantabrian*, Spain, 1992.
12. J. H. WILKINSON, The Algebraic Eigenvalue Problem, Oxford University Press, Oxford, 1965.