ALGORITHM PODQP
SOLVING POSITIVE DEFINITE QUADRATIC PROGRAMMING PROBLEMS

VASILE SIMA

Process Control System Laboratory
Research Institute for Informatics
8-10 Averescu Avenue
71316 Bucharest 1
ROMANIA

ABSTRACT

An efficient and numerically stable dual algorithm - PODQP - for solving positive definite quadratic programming problems is briefly described. Both inequality and equality constraints are dealt with. The algorithm employs an active set strategy and can exploit a priori information about an initial active set. In particular, the algorithm is strongly recommended for solving nonlinearly constrained optimization problems using projected Lagrangian techniques. Orthogonal transformations are used for updating the matrices and matrix factorizations after each change performed in the active set.

1. INTRODUCTION

The algorithm PODQP described in this paper solves positive definite (strictly convex) quadratic programming problems with equality and/or inequality constraints, mathematically defined in the form

$$\min_x \{c^T x + \frac{1}{2} x^T L L^T x \mid A_1^T x = b_1; A_2^T x \geq b_2\}, \tag{1}$$

where x and c are real n-vectors, $b \triangleq (b_1^T, b_2^T)^T$ is a real m-vector, L^T is an nxn real upper triangular nonsingular matrix, $A \triangleq (A_1, A_2)$ is a real nxm matrix, and b_1 and A_1 have m' elements and columns, respectively. Matrix L^T can be conceived as the Cholesky factor of the symmetric positive definite Hessian matrix of the problem, $G \triangleq LL^T$, and the i-th column of matrix A is the normal vector of the i-th constraint, $a_i^T x \geq b_i$, $i = 1:m$ (that is, i takes the values 1,2,...,m). We suppose that $n > 0$ and $m \geq m' \geq 0$. Note that equality constraints, if present, should be the first m' constraints.

Efficient and robust algorithms to solve (1) are primarily needed by the development of successive quadratic programming methods for solving general nonlinearly constrained optimization problems (Han, 1976; Powell, 1977, 1982). These highly successful methods require the solution of a strictly convex quadratic program to determine the search direction at each iteration k. The Hessian matrix of this program is a positive definite approximation G_k of the Hessian matrix of the Lagrangian function at the current iterate x_k. Usually, G_1 is set to the identity matrix and the Cholesky factorization of G_k is directly updated. This important usage of the algorithm to be described motivates the assumption on the availability of the Cholesky factor L^T. If L^T is not available, it can be obtained from G using the LINPACK subroutine DPOFA (Dongarra and co-workers, 1979).

The algorithm PODQP is based on a numerically stable dual algorithm proposed by Goldfarb and Idnani (1983), that is faster than other algorithms when an initial feasible point is not available. But PODQP includes several improvements. In the first place,

our algorithm uses more efficient solution techniques. For instance, Householder transformations are employed for updating the matrices after adding a new constraint to the active set. In addition, advantage can be taken of the prior information on an active set, as described in the next section. This is particularly useful when general nonlinearly constrained optimization problems are solved by successive quadratic programming methods, because then the active set tends not to change very much from one iteration to another and significant computational savings can be obtained.

Our code is an improvement of that listed in (Sima and Varga, 1986), and it can exploit the prior information about the active set. Several subroutines from the LINPACK package (Dongarra and co-workers, 1979) and IMSL library (*** IMSL, 1980) are also called. Details are given in the next section.

2. THE ALGORITHM

The algorithm does not need an initial feasible point, because it uses the dual feasibility. Starting from the easily computable unconstrained minimum of the problem (1), the algorithm solves a sequence of subproblems, maintaining the primal optimality, until primal feasibility is achieved. If the minimum step in the primal space - such that the *most violated* constraint (including the equality ones) does not violate the dual feasibility - is finite, then that constraint is added to the *working set*, that is the set of constraints satisfied as equalities. Otherwise, the nearest dual violated constraint is deleted (dropped) from the working set. (Clearly, the equality constraints are never candidates for deleting.) Only orthogonal transformations are used to update the quantities when the working set changes. The updates required when adding a constraint are made by a Householder transformation, using IMSL subroutine VHS12. The updates required when deleting a constraint are obtained by applying a sequence of plane rotations, computed by IMSL subroutine DROTG.

We now show in more details how the algorithm works. Clearly, an active set strategy is used. The solution of the problem (1) involves the solution of a sequence of quadratic programming subproblems, each having as working set a subset of constraints in (1). Let P(J) denote the subproblem with the objective function in (1) and with the constraints indexed by $J \subset M \triangleq \{1, 2, ..., m\}$. For instance, P(∅), where ∅ denotes the empty set, is the problem of finding the unconstrained minimum.

If the solution x of a subproblem P(J) is in the subspace defined by the active linearly independent constraints indexed by a set K in J, we say that (x,K) is a *subsolution*. Clearly, if (x,K) is a subsolution of the subproblem P(J), it is also a subsolution of the subproblem P(K). The algorithm for solving the positive definite quadratic programming problem (1) can be summarized as follows:

0) Assume a subsolution (x,K) be given.

1) Repeat, until all constraints are satisfied,

   1) Choose a violated constraint $p \in M \setminus K$.

   2) If $P(K \cup \{p\})$ has no feasible solutions, STOP (neither problem (1) has).

   3) Obtain a new subsolution $(\bar{x},\bar{K} \cup \{p\})$, where $\bar{K} \subset K$ and $F(\bar{x}) > F(x)$ and set $(x,K) \leftarrow (\bar{x},\bar{K} \cup \{p\})$.

2) STOP (x is the optimal solution for problem (1)).

Since the unconstrained minimum of the problem (1), $x_o = -G^{-1}c$ (with $G \triangleq LL^T$), can be very easily obtained, one may always initialize the above sketched algorithm with x_o. The previous implementations of this algorithm used this initialization. See, for instance, (Sima and Varga, 1986; Sima, 1990) in the context of solving general nonlinearly constrained optimization problems; more precisely, each quadratic programming subproblem encountered during the iterative process was solved by starting from the subsolution $(x_o,\emptyset)$. But the numerical experience has shown that the set of constraints that are active in the solution of the quadratic program tends not to change very much from one iteration to another. In many cases, the optimal basis remains the same for all iterations, and in other cases, there is an almost maximal intersection. To take advantage of this, the dual algorithm for solving the quadratic program can be called twice, except for the first iteration of the general algorithm for nonlinearly constrained optimization problems. On the first pass, all the constraints which are not in the optimal basis K of the previous iteration are ignored; in addition, the first violated constraint (with the least index) is chosen to be included into the working set, instead of the most violated one. On the second pass, all the constraints are taken into account, but the algorithm is initialized with the optimal solution of the first pass, instead of the unconstrained minimum. In this way, the number of working set changes reduces and an increase in efficiency is obtained. Note that the solution computed by the first pass is a subsolution of the current quadratic program, and the solution obtained after the second pass coincides with the solution of the current quadratic program. This strategy is implemented as an option in the Fortran code PODQP, described in the next section.

The implemented algorithm can be summarized as follows:

1° [Find the unconstrained minimum.]

If prior information is used, set J = K. Else set J = M.

$x = -G^{-1}c$, $F = c^Tx/2$, $H = G^{-1}$, $N^* = 0$, $K = \emptyset$, $t = 0$, $u = 0$.

2° [Choose a violated constraint, if exists.]

$s_j \triangleq a_j^Tx - b_j$, $j \in J \setminus K$. $V \triangleq \{j \in J \setminus K \mid s_j \neq 0, j \leq m', s_j < 0, j > m'\}$.

If $V = \emptyset$ and first pass, set J = M and go to 2° (begin the second pass).

Else if $V = \emptyset$, STOP (x is feasible and optimal).

Else, choose $p \in V$, and set $n = a_p$, $u \leftarrow (u^T,0)^T$.

3° [Check for feasibility and determine a new subproblem.]

3°1 [Determine step direction.]

$z = Hn$ (the step direction in the primal space).

If $t > 0$, $r = N^*n$ (the negative step direction in the dual space).

3°2 [Determine step length.]

If $r \leq 0$ or $t \leq m'$, then $\alpha_1 = \infty$.

Else, $\alpha_1 \triangleq u_l/r_l = \min \{u_j/r_j \mid r_j > 0, j = m'+1:t\}$ (the maximum step in the dual space without violating dual feasibility).

If $|z| = 0$, then $\alpha_2 = \infty$.

Else, $\alpha_2 \stackrel{\Delta}{=} -s_p/z^T n$ (the minimum step in the primal space such that the p-th constraint becomes feasible).

$\alpha = \min\{\alpha_1, \alpha_2\}$ (step length).

3°3 [Determine a new subproblem and take step.]

If $\alpha = \infty$, STOP (the subproblem and hence (1) are infeasible).

If $\alpha_2 = \infty$, then $u \leftarrow u + \alpha (-r^T, 1)^T$ (step in dual space),

$K \leftarrow K \setminus \{l\}$ (drop constraint l),

$t \leftarrow t - 1$, update H and N^*, and go to 3°1.

Else, $x \leftarrow x + \alpha z$, $F \leftarrow F + \alpha z^T n(\alpha/2 + u_{t+1})$,

$u \leftarrow u + \alpha(-r^T, 1)^T$ (step in primal and dual space).

If $\alpha = \alpha_2$ (full step), $K \leftarrow K \cup \{p\}$ (add constraint p),

$t \leftarrow t + 1$, update H and N^*, and go to 2°.

If $\alpha = \alpha_1$ (partial step), $K \leftarrow K \setminus \{l\}$ (drop constraint l),

$t \leftarrow t - 1$, update H and N^*, and go to 3°1.

In implementation, the matrices H and N^* are not used. The computations involving these matrices are replaced by other mathematically equivalent ones. More precisely, let $G = LL^T$ be the Cholesky factorization of G, and $L^{-1}N = Q\begin{bmatrix} R \\ O \end{bmatrix}$ be the QR factorization of $L^{-1}N$, where the columns of N are the t columns of A corresponding to the current active constraints. Denoting $J = L^{-T}Q = [J_1, J_2]$, where J_1 has t columns, then $H = J_2 J_2^T$, and $N^* = R^{-1}J_1^T$. Therefore, we have $z = J_2 d_2$, $r = R^{-1}d_1$, where $d \stackrel{\Delta}{=} J^T n$. The matrix Q is initialized to the identity matrix. All modifications caused by changes in the working set (adding and/or dropping constraints) are performed using orthogonal transformations (Householder transformations and plane rotations, respectively).

We also tried to replace the plane rotations by modified Householder transformations of order two. Theoretically, an improvement in efficiency would be expected, since the application of a plane rotation on two vectors of length n requires 4n floating-point multiplications and 2n additions, while the usage of modified Householder transformations involves 3n multiplications and additions. But no improvements have been obtained in our experiments on a FORMOX 286 (IBM-PC compatible) microcomputer, for vectors with elements randomly generated from a uniform distribution within the interval (0,1). Consequently, the plane rotations have been retained in our codes.

3. USAGE

The algorithm briefly described above is implemented in the subroutine PODQP, to be presented in this section. The basic information on the usage of this subroutine, listed below, is directly obtained from the first lines of its code, written in the Fortran 77 language.

```
      SUBROUTINE PODQP(N, MEQ,M,C,G,LG,A,LA,B,T,ACTIVE,X,
     1           FVAL,VLAM,INFO,W,LW,R,LR,WA1,WA2)
C
C  PODQP subroutine solves a positive definite quadratic programming
C  problem with N variables and M constraints, defined by
C    min (C'X + (1/2)X'GX ! A'X .GE. B),   G = LL',
C  where the Cholesky factor of G, L', is given. (' denotes the
C  transposition.) The dual algorithm of Goldfarb and Indani, with
C  extensions to solve problems including equality constraints and
C  to take into account the prior information about the active set,
C  is implemented.
C
C  Declaration of formal arguments.
C
      INTEGER              INFO,LA,LG,LR,LW,M,MEQ,N,T
      INTEGER              ACTIVE(*)
      DOUBLE PRECISION FVAL
      DOUBLE PRECISION A(LA,*),B(*),C(N),G(LG,N),R(LR,*),VLAM(*),
     1                     W(LW,N),WA1(N),WA2(*),X(N)
C
C  Description of arguments.
C
C  N is a positive integer input variable that specifies the number
C    of independent variables.
C
C  MEQ is a non-negative integer input variable that specifies the
C    number of equality constraints.
C
C  M is a non-negative integer input variable that specifies the
C    number of constraints (including the equality constraints).
C    M should be greater than or equal to MEQ.
C
C  C is a real one-dimensional array of dimension at least N that
C    contains on input the coefficients of the linear terms of the
C    objective function of the problem. C is unchanged on output.
C
C  G is a real two-dimensional array with row dimension LG and
C    column dimension at least N. On input, the N by N upper
C    triangular part of G contains the Cholesky factor L' of the
C    matrix G. The strict lower part is arbitrary. G is unchanged
C    on output.
C
C  LG is a positive integer input variable that specifies the first
C    dimension of the array G. LG should be greater than or equal
C    to N.
C
C  A is a real two-dimensional array with row dimension LA and
C    column dimension at least M that contains the transpose of the
C    constraint matrix. If M = 0, A is not referenced. If MEQ > 0,
C    the first MEQ columns of A correspond to equality constraints.
C    A is unchanged on output.
C
C  LA is a positive integer input variable that specifies the first
C    dimension of the array A. LA should be greater than or equal
C    to N.
```

```
C
C  B is a real one-dimensional array of dimension at least M that
C   contains on input the coefficients of the right hand side of the
C   constraints of the problem. If M = 0, B is not referenced.
C   B is unchanged on output.
C
C  T is a non-negative integer input/output variable that contains
C   the number of constraints which are active at the solution.
C   T is to be input only if INFO .NE. 0 on input.
C
C  ACTIVE is an integer one-dimensional array of dimension at least
C   M, if on input INFO = 0, and 2*M, if on input INFO .NE. 0.
C   If INFO .NE. 0, on input, then the first T elements of ACTIVE
C   contain the indices of the constraints which are assumed to be
C   active at the algorithm initialization. If INFO = 0, on input,
C   then the content of ACTIVE is arbitrary. On output, ACTIVE(J),
C   J = 1,2,...,T, contains the index of the J-th active constraint.
C
C  X is a real one-dimensional array of dimension at least N that
C   contains on output the final estimate of the solution.
C
C  FVAL is a real output variable that contains the value of the
C   objective function at the solution X.
C
C  VLAM is a real one-dimensional array of dimension at least M that
C   contains on output the Lagrange multipliers at the solution X.
C   The multipliers corresponding to the non-active constraints
C   are set to zero.
C
C  INFO is an integer input/output variable that contains an option/
C   /error diagnosis information.
C   On input, the values of INFO and their significance are:
C     INFO = 0, No prior information is available.
C     INFO # 0, Prior information is available in T and ACTIVE.
C           In this case, two passes through Goldfarb-Idnani
C           algorithm are performed.
C   On output, the values of INFO and their significance are:
C     INFO = -I, I = 1,...,N, The diagonal element G(I,I) is zero.
C     INFO = 0, Invalid input parameters.
C     INFO = 1, Solution has been determined.
C     INFO = 2, Problem constraints seem to be inconsistent.
C     INFO = 3, T is greater than N, but X is infeasible.
C
C  W is a real two-dimensional working array with row dimension LW
C   and column dimension at least N. During the computations,
C   W contains the matrix J in the Goldfarb-Idnani algorithm.
C
C  LW is a positive integer input variable that specifies the first
C   dimension of the array W. LW should be greater than or equal
C   to N.
C
C  R is a real two-dimensional working array with row dimension LR
C   and column dimension at least N + 1. During the computations, the
C   upper triangular part of array R contains the upper triangular
C   matrix R in the Goldfarb-Idnani algorithm. The subdiagonal
```

```
C     locations of R are used as working storage.
C
C   LR is a positive integer input variable that specifies the first
C     dimension of the array R. LR should be greater than or equal
C     to N.
C
C   WA1 and WA2 are real one-dimensional working arrays with dimension
C     at least N and max(N,M), respectively. WA1 and WA2 temporarily
C     contain the vectors z and r in the Goldfarb-Idnani algorithm.
C
C   Called subprograms
C
C     D1MACH (PORT library)
C     DDOT, DROTG, VHS12 (IMSL library)
C     DTRSL (LINPACK package)
C
```

Subroutine PODQP is useful for solving any positive definite quadratic programming problem, when an initial feasible point is not known.

4. PERFORMANCES

We shall merely report here the results obtained with the latest version of the code, since numerical results for the previous version were implicitly presented in (Sima and Varga, 1986; Sima, 1990). Consequently, we shall compare the gain in speed resulted from the usage of prior information on the active set when solving general nonlinearly constrained optimization problems with Powell's method (Powell, 1982), which involves the solution of a positive definite quadratic programming subproblem at each iteration.

Table 1 presents the execution times in seconds on a FORMOX 286 microcomputer (MS DOS operating system, Fortran compiler, version 5.00), for solving several nonlinear problems with both the old subroutine (PODPP) and the new one. We mention that all increase in speed is due to the usage of the information about the active set from an iteration to the following one. The number of quadratic programming subproblems, as well as the total number of additions/deletions of constraints to/from the working set are also given.

The characteristics of the problems in Table 1 are defined in Table 2, where references for the mathematical formulation are also given.

Note that we solved 127 quadratic programming problems. The total number of constraint additions to the working set was 1082 for the old version (PODPP) and 963 for the new version (PODQP). The total number of constraint deletions was 140 for the old version (PODPP) and 21 for the new version (PODQP). Note that the total number of constraint deletions never exceeded 3 for the new version, while the old version sometimes involved 24 deletions. The number of constraint additions was also reduced accordingly. The mean value of the gain in efficiency is 23 % (if an average of the percentages in Table 1 is computed), but the actual time gain is even greater (about 25 %), since small gains were obtained for easier to solve problems.

TABLE 1. Comparative performance results when solving nonlinearly constrained optimization problems.

Problem No	Time (seconds) with PODPP	Time (seconds) with PODQP	Gain (%)	No. subproblems	No. additions PODPP/PODQP	No. deletions PODPP/PODQP
1	1.10	0.93	15	6	24/24	0/0
2	1.59	1.37	14	10	40/39	1/0
3	4.56	3.07	33	14	63/43	21/1
4	7.96	5.11	36	7	80/63	20/3
5	1.10	0.94	15	7	24/24	0/0
6	12.64	10.16	20	6	84/80	7/3
7	7.58	5.16	32	3	52/42	10/0
8	28.17	20.43	27	12	184/162	24/2
9	6.37	5.33	16	3	42/40	5/3
10	27.90	21.92	21	13	182/171	14/3
11	26.15	18.95	28	11	170/150	23/3
12	1.65	1.43	13	12	35/32	4/1
13	6.37	4.88	23	23	102/93	11/2

TABLE 2. Characteristics of and references to the problems in Table 1.

Problem No.	Dimensions n	m'	m	Active constr. t	References and initialization x_0
1	4	1	10	3	Bartholomew-Biggs, 1978, $x_0 = (1,5,5,1)^T$.
2	4	1	5	3	Lowe, 1974; Sima and Varga, 1986 (Ex. 6.4.1), $x_0 = (3,.5,.5,3)^T$.
3	6	0	8	2	Bartholomew-Biggs, 1978; Sima and Varga, 1986 (Ex. 6.4.2), $x_0 = (5.54,4.4,12.02,11.82,.702,.852)^T$.
4	9	6	14	8	Bartholomew-Biggs, 1978; Sima and Varga, 1986 (Ex. 6.4.3), $x_0 = (.8,.8,.2,.2,1.0454,1.0454,1.0454,0,0)^T$.
5	4	1	8	3	Dumitru, 1975; Sima and Varga,1986 (Ex.6.4.6), $x_0 = (.4,.22,.13,.25)^T$.
6	13	1	17	13	Lowe, 1974; Sima and Varga, 1986 (Ex.6.4.7, linear model), $x_0 = (.2,.1,.1,.1,.1,.3,.1,.1,.2,.1,.1,.1,.1)^T$.
7	13	2	17	13	Lowe, 1974; Sima and Varga, 1986 (Ex. 6.4.7, nonlinear model), $x_0 = (1,.1,.1,.1,.2,.1,.1,.1,.1,.1,.1,.1,.1)^T$.
8	13	1	17	12	idem, x_0 as for problem 7.
9	13	1	17	12	idem, $x_0 = (.4,.05,.05,.05,.05,.3,.01,.01,.1,.03,.03,.03,.2)^T$.
10	13	1	17	12	idem, x_0 as for problem 9, greater accuracy.
11	13	1	17	12	idem, x_0 = = $(.785587,0,0,0,0,.173919,0,0,.020643,0,0,0,.019853)^T$.
12	4	0	6	2	Test example for NAG library, $x_0 = (3,-1,0,1)^T$.
13	5	1	12	3	Test example for NAG library (NAG, 1981), $x_0 = (.5,1,.6,1.4,1)^T$.

5. CONCLUSIONS

A new implementation of the dual algorithm of Goldfarb and Idnani (1983), for solving strictly convex quadratic programming problems, is presented. It can take into account the prior knowledge about the initial active set. This option is particularly useful when solving general nonlinearly constrained optimization problems using successive quadratic programming techniques. A significant gain in efficiency is obtained in this case, as the numerical experience illustrates. The algorithm can also be modified to exploit the prior knowledge about an initial feasible point, and the corresponding primal-dual algorithm will be suitable especially for linearly constrained optimization problems.

REFERENCES

[1] BARTHOLOMEW-BIGGS, M.C., **A numerical comparison between two approaches to the nonlinear programming problem.** In L.C.W. Dixon, G.P. Szegö (Eds.), Toward Global Optimization, North-Holland Publishing Co., Amsterdam, New York and Oxford (1978), pp. 293-312.

[2] DUMITRU, V., **Nonlinear Programming** (in Romanian). Editura Academiei R.S.R., Bucuresti (1975).

[3] DONGARRA, J.J., BUNCH, J.R. MOLER, C.B. STEWART, G.W. **LINPACK Users Guide**, SIAM Publications, Philadelphia (1979).

[4] GOLDFARB, D., IDNANI, A., **A numerically stable dual method for solving strictly convex quadratic programs.** MATH. PROG. 27 (1983), pp. 1-33.

[5] HAN, S-P., **Superlinearly convergent variable metric algorithms for general nonlinear programming problems.** MATH. PROG. 11 (1976), pp. 263-282.

[6] *** IMSL, **IMSL Library Reference Manual.** IMSL Inc., Houston, Texas (1980).

[7] LOWE, M., **Nonlinear Programming: Augmented Lagrangean Techniques for Constrained Minimization.** Ph.D. Thesis, Montana State University (1974).

[8] *** NAG **NAG Fortran Library Reference Manual** (Mark 8). Numerical Algorithms Group Limited, Oxford (1981).

[9] POWELL, M.J.D., **A fast algorithm for nonlinearly constrained optimization calculations.** DAMTP 77/NA2 Report (1977), University of Cambridge.

[10] POWELL, M.J.D., **VMCWD : A Fortran subroutine for constrained optimization.** DAMTP 82/NA4 Report (1982), University of Cambridge.

[11] POWELL, M.J.D., **On the quadratic programming algorithm of Goldfarb and Idnani.** MATH. PROG. 25 (1985), pp. 46-61.

[12] POWELL, M.J.D., **Updating conjugate directions by the BFGS formula.** MATH. PROG. 38 (1987), pp. 29-46.

[13] SIMA, V., **A software package for constrained optimization with applications to control problems.** Studies and Researches in Computers and Informatics, ICI, 3 (1990), pp. 117-127.

[14] SIMA, V., VARGA, A., **Computer Aided Optimization Practice** (in Romanian). Editura Tehnica, Bucuresti (1986).