QL: A Fortran Code for Convex Quadratic Programming
- User's Guide -

K. Schittkowski

Address: Prof. K. Schittkowski
Siedlerstr. 3
D - 95488 Eckersdorf
Germany

Phone: (+49) 921 32887

E-mail: klaus@schittkowski.de

Web: http://www.klaus-schittkowski.de

Date: February, 2011

Abstract

The Fortran subroutine QL solves strictly convex quadratic programming problems subject to linear equality and inequality constraints by the primal-dual method of Goldfarb and Idnani. An available Cholesky decomposition of the objective function matrix can be provided by the user. Bounds are handled separately. The code is designed for solving small-scale quadratic programs in a numerically stable way. Its usage is outlined and an illustrative example is presented.

Keywords: quadratic optimization, QP, quadratic programming, numerical algorithms, Fortran codes

1 Introduction

The code solves the strictly convex quadratic program

$$\begin{array}{rll} & \min \ \frac{1}{2}x^T C x + d^T x & \\ & a_j^T x + b_j = 0 \quad , & j = 1, \ldots, m_e \ , \\ x \in I\!R^n : & a_j^T x + b_j \geq 0 \quad , & j = m_e + 1, \ldots, m \ , \\ & x_l \leq x \leq x_u & \end{array} \tag{1}$$

with an n by n positive definite matrix C, an n-dimensional vector d, an m by n matrix $A = (a_1, ..., a_m)^T$, and an m-vector b. Lower and upper bounds for the variables, x_l and x_u, respectively, are separately handled.

The quadratic program (1) is solved by the primal-dual method of Goldfarb and Idnani [2]. Initially, a Cholesky decomposition of C is computed by an upper triangular matrix R such that $C = R^T R$. If available, a user can provide a known triangular factor. In case of numerical instabilities, e.g. round-off errors, or a semi-definite matrix C, a certain multiple of the unit matrix is added to C to get a positive definite matrix for which a Cholesky decomposition can be obtained.

Successively, violated constraints are added to an *active set* until a solution is obtained. In each step, the minimizer of the objective function subject to the new active set is computed. If an iterate satisfies all linear constraints and bounds, the optimal solution is obtained and the the algorithm terminates. If necessary, a constraint can be dropped from the working set if no longer considered as an active one.

The corresponding matrix manipulations to compute the successive minimizers subject to active constraints are performed in a numerically stable way by orthogonal Givens rotations. Since objective function values are strictly increasing from one iteration to the next and since the number of possible active sets is finite, the algorithm terminates after finitely many steps.

A particular advantage of a dual method is that the phase I of a primal algorithm, i.e., the computation of an initial feasible point satisfying all linear constraints and bounds, can be avoided. A generalization of the method introduced in this section, is published by Boland [1] for the case that C is positive semi-definite.

The implementation of the code goes back to Powell [4] and the algorithmic details are found in the reference. Besides of a few internal changes concerning numerical details, the main extensions of QL compared to ZQPCVX are the separate handling of upper and lower bounds and the optional provision of the Cholesky factor of C.

As part of the sequential quadratic programming code NLPQL for constrained nonlinear programming, QL is frequently used in practice. Even many of the standard test problems of the collections of Hock and Schittkowski [3] and Schittkowski [6] are badly scaled, ill-conditioned, or even degenerate. The reliability of an SQP solver depends mainly on the numerical efficiency and stability of the code solving the quadratic programming subproblem. As verified by the comparative study of Schittkowski [10], NLPQL and QL successfully

solve all 306 test problems under consideration. In addition, all 1,000 test examples of the interactive data fitting system EASY-FIT, see Schittkowski [9], are solved by the code DFNLP. The subroutine is an extension of NLPQL and depends also on the stable solution of quadratic programming subproblems, see Schittkowski [7].

2 Program Documentation

The Fortran subroutine QL reorganizes some data to solve the quadratic program (1) by a modification of a code going back to Powell [4]. An extension is available for special subproblems arising when solving nonlinear least squares problems, see Schittkowski [7].

Usage:

```
      CALL  QL(     M,    ME,  MMAX,     N,  NMAX,
     /              MNN,   C,    D,     A,     B,
     /              XL,    XU,   X,     U,     EPS,
     /              MODE,  IOUT, IFAIL, IPRINT, WAR,
     /              LWAR,  IWAR, LIWAR          )
```

Declaration of arguments:

M :	Number of constraints.
ME :	Number of equality constraints.
MMAX :	Row dimension of array A containing linear constraints. MMAX must be at least one and greater or equal to M.
N :	Number of optimization variables.
NMAX :	Row dimension of C. NMAX must be at least one and greater or equal to N.
MNN :	Must be equal to M+N+N when calling QL, dimension of U.
C(NMAX,N) :	Objective function matrix which should be symmetric and positive definite. If MODE=0, C is supposed to be the upper triangular factor of a Cholesky decomposition of the objective function matrix.

D(N) : Contains the constant vector of the quadratic objective function.

A(MMAX,N) : Matrix of the linear constraints, first ME rows for equality, then M-ME rows for inequality constraints.

B(M) : Constant values of linear constraints in the same order.

XL(N),XU(N) : On input, the one-dimensional arrays XL and XU must contain the upper and lower bounds of the variables.

X(N) : On return, X contains the optimal solution.

U(MNN) : On return, U contains the multipliers subject to the linear constraints (first M locations) and bounds. At the optimal solution, all multipliers of inequality constraints are nonnegative.

EPS : Final termination accuracy (e.g. 1.0D-12). The parameter value should not be smaller than the underlying machine precision.

MODE :
- MODE=0 - Initial Cholesky factorization of C provided by the calling program, stored in the upper triangular part of array C.
- MODE=1 - Cholesky decomposition internally computed.

IOUT : Output unit number, i.e., all write-statements start with 'WRITE(IOUT,... '.

IFAIL : Termination reason:
- IFAIL=0 : Optimality conditions satisfied.
- IFAIL=1 : Termination after too many iterations (40*(N+M)).
- IFAIL=2 : Termination accuracy insufficient.
- IFAIL=3 : Inconsistency, division by zero.
- IFAIL=4 : Numerical instabilities.
- IFAIL=5 : LWAR, LIWAR, MNN, or EPS incorrect.
- IFAIL>100 : Inconsistent constraints and IFAIL=100+ICON, where ICON denotes a constraint causing the conflict.

IPRINT : Specification of the desired output level:

	IPRINT=0 : No output of the program.
	IPRINT=1 : Only a final error message is given.
WAR(LWAR) :	Real working array of length LWAR. If QL is terminated with IFAIL=4, rounding errors prevent satisfaction of constraints and QL can be restarted with EPS=WA(1).
LWAR :	Length of WAR, must be at least 3*NMAX*NMAX/2 + 10*NMAX + 2*MMAX + 1.
IWAR(LIWAR) :	Integer working array of length LIWAR.
LIWAR :	Length of IWAR, should be at least N.

3 Example

To give an example how to organize the code, we consider a simple quadratic program,

$$\begin{aligned} & \min \ \tfrac{1}{2}\sum_{i=1}^{5} x_i^2 - 21.98x_1 - 1.26x_2 + 61.39x_3 + 5.3x_4 + 101.3x_5 \\ x_1, \ldots x_5 \in I\!R: \ & -7.56x_1 + 0.5x_5 + 39.1 \geq 0 \ , \\ & -100 \leq x_i \leq 100 \ , \quad i = 1 \ldots, 5 \end{aligned}$$

The corresponding Fortran source code is listed below.

```
      IMPLICIT NONE
      INTEGER           NMAX, MMAX, NXMNN, LWAR, LIWAR
      PARAMETER        (NMAX   = 5,
     /                  MMAX   = 1,
     /                  NXMNN  = MMAX + NMAX + NMAX,
     /                  LWAR   = 3*NMAX*NMAX/2 + 10*NMAX + 2*MMAX + 1,
     /                  LIWAR  = NMAX)
      INTEGER           N, M, ME, MODE, IOUT, IPRINT, IFAIL, MNN, I, J,
     /                  IWAR(LIWAR)
      DOUBLE PRECISION X(NMAX), A(MMAX,NMAX), B(MMAX), U(NXMNN),
     /                  XL(NMAX), XU(NMAX), C(NMAX,NMAX), D(NMAX),
     /                  WAR(LWAR), EPS, F, T
C
C   Set some constants
C
      IOUT   = 6
      IPRINT = 1
      EPS    = 1.0D-12
      N      = 5
      M      = 1
      ME     = 0
```

```
      MNN    = M + N + N
C
C   Set problem data
C
      DO I=1,M
         DO J=1,N
            A(I,J) = 0.0D0
         ENDDO
         B(I) = 0.0D0
      ENDDO
      DO I=1,N
         DO J=1,N
            C(I,J) = 0.0D0
         ENDDO
         C(I,I) =    1.0D0
         D(I)   =    0.0D0
         XL(I)  = -100.0D0
         XU(I)  =  100.0D0
      ENDDO
      A(1,1) =  -7.56D0
      A(1,5) =   0.5D0
      B(1)   =  39.1D0
      D(1)   = -21.98D0
      D(2)   =  -1.26D0
      D(3)   =  61.39D0
      D(4)   =   5.3D0
      D(5)   = 101.3D0
C
C   Execute QL with predetermined Cholesky decomposition
C
      MODE = 0
      CALL QL (      M,     ME,   MMAX,      N,   NMAX,
     /             MNN,      C,      D,      A,      B,
     /              XL,     XU,      X,      U,    EPS,
     /            MODE,   IOUT,  IFAIL, IPRINT,    WAR,
     /            LWAR,   IWAR,  LIWAR)
C
C   Final objective function value
C
      F = 0.0D0
      IF (IFAIL.EQ.0) THEN
         DO I=1,N
            T = 0.0D0
            DO J=1,N
```

```
               T = T + C(I,J)*X(J)
            ENDDO
            F = F + (0.5D0*T + D(I))*X(I)
         ENDDO
      ENDIF
C
C   Output generation
C
         WRITE(IOUT,*) 'Optimal solution values:'
         WRITE(IOUT,*) (X(I),I=1,N)
         WRITE(IOUT,*) 'Objective function value:',F
C
C   End of main program
C
      STOP 'QL_DEMO'
      END
```

The following output should appear on the screen:

```
 Optimal solution values:
  -1.42539840706855        1.26000000000000       -61.3900000000000
  -5.30000000000000       -99.7520239148764
 Objective function value:  -6996.50559772314
```

Some further quadratic programs for testing a correct implementation are found in Hock and Schittkowski [3].

References

[1] Boland N.L. (1997): *A dual-active-set algorithm for positive semi-definite quadratic programming*, Mathematical Programming, Vol. 78, 1-27

[2] Goldfarb D., Idnani A. (1983): *A numerically stable method for solving strictly convex quadratic programs*, Mathematical Programming, Vol. 27, 1-33

[3] Hock W., Schittkowski K. (1981): *Test Examples for Nonlinear Programming Codes,* Lecture Notes in Economics and Mathematical Systems, Vol. 187, Springer

[4] Powell M.J.D. (1983): *ZQPCVX, A FORTRAN subroutine for convex quadratic programming*, Report DAMTP/1983/NA17, University of Cambridge, England

[5] Schittkowski K. (1985/86): *NLPQL: A Fortran subroutine solving constrained nonlinear programming problems,* Annals of Operations Research, Vol. 5, 485-500

[6] Schittkowski K. (1987a): *More Test Examples for Nonlinear Programming,* Lecture Notes in Economics and Mathematical Systems, Vol. 182, Springer

[7] Schittkowski K. (1988): *Solving nonlinear least squares problems by a general purpose SQP-method,* in: Trends in Mathematical Optimization, K.-H. Hoffmann, J.-B. Hiriart-Urruty, C. Lemarechal, J. Zowe eds., International Series of Numerical Mathematics, Vol. 84, Birkhäuser, 295-309

[8] Schittkowski K. (1999): *EASY-OPT: An interactive optimization system with automatic differentiation - User's guide*, Report, Department of Mathematics, University of Bayreuth, D-95440 Bayreuth

[9] Schittkowski K. (2002): *Numerical Data Fitting in Dynamical Systems*, Kluwer Academic Publishers, Dordrecht

[10] Schittkowski K. (2006): *NLPQLP: A Fortran implementation of a sequential quadratic programming algorithm with distributed and non-monotone line search - User's guide, version 2.2*, Report, Department of Computer Science, University of Bayreuth, D-95440 Bayreuth