User Information for `bqpd`

Introduction

Welcome to `bqpd`! This is a package of Fortran 77 subroutines for solving quadratic programming (QP) problems in the form

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T G\mathbf{x} \\ \text{subject to} & \mathbf{l} \leq [I \,:\, A]^T\mathbf{x} \leq \mathbf{u} \end{array}$$

where $\mathbf{x}$ and $\mathbf{c}$ are n-vectors, G is a symmetric $n \times n$ matrix (the Hessian matrix), and A is an $n \times m$ matrix. Thus the first n constraints are simple bounds, and the remaining m constraints are general linear constraints whose normal vectors are columns of A.

More specifically, the package is designed to find a Kuhn–Tucker (KT) point of this problem (see for example R. Fletcher, *Practical Methods of Optimization, 2nd. Edition*, John Wiley, 1987, for background information). If G is positive semi-definite then the KT point is a global solution, else usually a local solution. The method may also be used efficiently to solve a linear programming (LP) problem ($G = 0$). A recursive form of a null-space active set method is used, using Wolfe's method to resolve degeneracy. This would guarantee termination if exact arithmetic could be used. Matrix information is made available and processed by calls to external subroutines. Details of these are given in an auxiliary file named either `denseL.f` or `sparseL.f`.

The `bqpd` package provides a number of features which make for flexible use, and reliable and efficient solution. These include

- indefinite Hessian allowed
- two-sided simple bounds and general constraints
- sparse or dense format for $\mathbf{c}$ and A
- sparse or dense matrix algebra modules
- cold/warm/hot starts
- steepest-edge pivoting
- automatic scaling option
- efficient as an LP solver
- effective in both single and double precision
- new features for numerical stability.

Using `bqpd`

The master subroutine is stored in the file `bqpd.f`, and other auxiliary files store subroutines used by `bqpd.f`. To use `bqpd`, a routine which drives (i.e. calls) `bqpd` must be supplied. Examples of driver routines are supplied which will drive `bqpd.f` from a data file. To use `bqpd` with dense matrix storage, the driver program `ddd.f` (or the user's driver program) and the files

`bqpd.f`, `denseL.f`, `denseA.f`, `auxil.f`, `util.f`

must be compiled and linked. To use `bqpd` with sparse matrix storage, the driver program `sss.f` and the files

`bqpd.f`, `sparseL.f`, `sparseA.f`, `auxil.f`, `util.f`

must be compiled and linked. The combination of `denseL.f` and `sparseA.f` is also allowed. The subroutines are written as though for single precision (r4) usage. Users with SUN equipment can obtain double precision (r8) arithmetic by compiling with the `-r8` flag. Other users should globally change all occurrences of the word `REAL` to `double precision` and `.E` to `.D` in all the files supplied. It is also recommended to change `DOUBLE PRECISION` to `real*16` (or similar) throughout, whenever your system supports quadruple precision. In this case it is advisable (although probably not imperative) to change the use of `dble(.)` to some suitable function for conversion to `real*16` type. In SUN Fortran the appropriate function is `qext(.)`. Changing from single to double precision is recommended for large or ill-conditioned problems. IT IS IMPORTANT TO SET THE DEFAULT TOLERANCES (around line 1175 of `bqpd.f`) TO MATCH THE PRECISION BEING USED.

Data structures for specifying the QP problem

The vectors $\mathbf{x}$, $\mathbf{l}$ and $\mathbf{u}$ are stored as dense vectors in the usual Fortran 77 form. Equality constraints are designated by having $l_i = u_i$. The matrix $[\mathbf{c} : A]$ may be stored in either dense or sparse format. The dense format is the conventional Fortran 77 storage scheme for a matrix, that is by columns, with a possible gap between each column. The 'stride' between columns is stored in the parameter `la` of `bqpd.f` (set `la` to n for no gap between the columns).

The sparse format is explained in the header of file `sparseA.f`. It may be illustrated for the `avgas` test problem, for which $n = 8$, $m = 10$,

$$[\mathbf{c} : A] = \begin{bmatrix} 0 & -1 & & & & -1 & & 2 & 5 & & \\ -2 & -1 & & & & & -1 & & & 1 & 1 \\ -1 & & -1 & & & -1 & & 1 & 3 & & \\ -3 & & -1 & & & & -1 & & & -1 & \\ -2 & & & -1 & & -1 & & & -3 & & \\ -4 & & & -1 & & & -1 & & & -3 & -3 \\ -3 & & & & -1 & -1 & & -1 & -1 & & \\ -5 & & & & -1 & & -1 & & & -5 & -2 \end{bmatrix},$$

$$\mathbf{l}^T = (0,\, 0,\, 0,\, 0,\, 0,\, 0,\, 0,\, 0, -1, -1, -1, -1, -2, -2,\, 0,\, 0,\, 0,\, 0)$$

and

$$\mathbf{u}^T = (1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1, \infty, \infty, \infty, \infty, \infty, \infty, \infty, \infty, \infty, \infty).$$

In sparse format, the nonzero elements of $[\mathbf{c},\ A]$ would be stored in the vector

$$\texttt{a} = (-2, -1, -3, -2, -4, -3, -5; -1, -1; -1, -1; -1, -1; -1, -1; -1, -1, -1, -1; \ldots$$

$$\ldots -1, -1, -1, -1;\ 2,\ 1, -1;\ 5,\ 3, -3, -1;\ 1, -1, -3, -5;\ 1, -3, -5)$$

and the indexing information in the vector `la(0:*)` would be

$$(38 \,|\, 2, 3, 4, 5, 6, 7, 8;\ 1, 2;\ 3, 4;\ 5, 6;\ 7, 8;\ 1, 3, 5, 7; \ldots$$

$$\ldots\ 2, 4, 6, 8;\ 1, 3, 7;\ 1, 3, 5, 7;\ 2, 4, 6, 8;\ 2, 6, 8 \,|\, 1, 8, 10, 12, 14, 16, 20, 24, 27, 31, 35, 38).$$

Data files for the avgas problem, suitable for dense and sparse formats respectively, are supplied in the files `avgas.d` and `avgas.s`.

The Hessian matrix G is supplied indirectly to `bqpd.f` by means of a user subroutine `gdotx` which calculates the vector $\mathbf{v} = G\mathbf{x}$ from a given vector $\mathbf{x}$. The format of this subroutine is explained in the header to the file `bqpd.f`. Information for use with `gdotx` is supplied to the subroutine `bqpd` via the parameters `ws` and `lws`. Thus the user is free to specify G in whatever way is most convenient.

Setting and Interpreting the Parameters of `bqpd`

The definition of the parameters of `bqpd` is explained in the header to the file `bqpd.f`. Most of this is straightforward but some further explanation of how to interpret the contents of the parameters `ls` and `r` may be helpful. This is done by reference to the `avgas` test problem above, the solution of which is

$$\mathbf{x}^T = (0,\ 1,\ \tfrac{3}{4},\ \tfrac{1}{4},\ \tfrac{1}{2},\ \tfrac{1}{4},\ \tfrac{3}{4},\ 0).$$

In examining the outcome of `bqpd`, we first inspect the resulting vector

$$\texttt{ls} = (8,\ 15, -2,\ 9,\ 10,\ 16,\ 17,\ 13,\ 18,\ 4,\ 11, -3,\ 5,\ 14, -7,\ 12,\ \ 6,\ 1)$$

and the parameter values $\texttt{k} = 0$, $\texttt{peq} = 0$, $\texttt{lp(1)} = 1$ and $\texttt{lp(2)} = 9$, which determine how `ls` is partitioned. We recall that constraint indices in the range 1:8 (that is 1:n) refer to simple bounds, and those in the range 9:18 to general constraints. We deduce that there are no active equality constraints or pseudo-bounds, since the ranges 1:`peq` and `peq`+1:`lp(1)`–1 are empty. (Had `peq` been say 2, it would be deduced from `ls` that constraints 8 and 15 were active equality constraints.) Thus the active set constraints at the solution are those in positions 1:8 (that is `lp(1)`:n–k) of `ls`. The negative sign on $\texttt{ls(3)} = -2$ indicates that the upper bound of constraint 2 (that is $x_2 \leq 1$) is active. Constraint indices in positions 9:18 (that is `lp(2)`:n+m) of `ls` determine the inactive constraints, or more precisely the constraints not in the active set. The sign of these indices indicates which bound is closest to being active.

The contents of the vector

$$\texttt{r} = (0,\ \tfrac{5}{6},\ \tfrac{1}{4},\ \tfrac{1}{4},\ \tfrac{1}{2},\ \tfrac{1}{4},\ \tfrac{1}{4},\ \tfrac{5}{3},\ \tfrac{5}{2},\ \tfrac{5}{3},\ \tfrac{1}{4},\ \tfrac{1}{4},\ \tfrac{11}{8},\ \tfrac{1}{2},\ \tfrac{17}{12},\ \tfrac{5}{24},\ \tfrac{4}{3},\ \tfrac{1}{4})$$

give the multipliers of the active constraints and the residuals of the inactive constraints, stored in the natural ordering. Thus $\texttt{r(2)} = \frac{5}{6}$ indicates that the multiplier of the constraint $x_2 \leq 1$ is $\frac{5}{6}$, since this constraint is in the active set. Also, since constraint -7 is an inactive constraint, the value $\texttt{r(7)} = \frac{1}{4}$ indicates that the residual of this constraint, measured from its upper bound, is $\frac{1}{4}$. Likewise the residual, or slack, in constraint 18, which is also inactive, is $\frac{1}{4}$, measured from its lower bound. The sign convention for `r` is that its elements are always nonnegative at a KT point (except possibly for multipliers of equality constraints). Note that constraint 1, which is not in the active set, has a residual $\texttt{r(1)} = 0$, indicating that it is active but degenerate.

Post-processing with `bqpd`

`bqpd` is a null-space method, for which we shall follow the notation of Chapter 10.1 of *Practical Methods of Optimization* referenced above. The matrix V is made up of unit vectors as in Equation (10.1.21), chosen so as to keep the null-space matrix Z well-conditioned. The indices of the unit vectors are stored in positions `ll`+1:`ll`+k of the vector `lws`. Operations with Z or Z^T can be implemented by making solves with the basis matrix $B = [A : V]$. Subroutines `fbsub` and `tfbsub` which enable this to be done are provided with a common interface by both `denseL.f` and `sparseL.f`, and are documented in the headers to these subroutines.

It may also be required to carry out post-processing using the reduced Hessian matrix Z^TGZ at the solution. In `bqpd` the Choleski factor R, given by $Z^TGZ = R^TR$ is stored starting at location `kk`+1 of the vector `ws`. Utilities to perform solves with the reduced Hessian are provided by subroutines `rsol` and `rtsol` in the file `util.f`. The data structure of R and the mode of use of these subroutines is described in their headers. Note that the reduced Hessian is always stored in dense format, irrespective of whether a dense or sparse format is used for storage of A or factors of B. A consequence of this is that `bqpd` becomes inefficient if the dimension k of the null-space becomes very large.

The parameters `kk` and `ll` in the above description are those set by the user in `common/wsc/` (see the header to `bqpd`).

R. Fletcher, April 1998