IMM
DEPARTMENT OF MATHEMATICAL MODELLING

Technical University of Denmark
DK-2800 Lyngby – Denmark

J. No. QP3
14.4.1997
HBN/ms

QPBOX USER'S GUIDE

A Fortran77 Package for Box Constrained Quadratic Programs

Hans Bruun Nielsen

TECHNICAL REPORT
IMM-REP-1997-04

IMM

Contents

1. Introduction

Let $\boldsymbol{c} \in I\!R^n$ and $\boldsymbol{H} \in I\!R^{n\times n}$ be a given vector and symmetric, positive definite matrix, respectively. The subroutine `QPBOX` finds the solution y^* to the quadratic programming problem

$$\begin{aligned} &\min_{\boldsymbol{y}}\{q(\boldsymbol{y}) \equiv \tfrac{1}{2}\boldsymbol{y}^T\boldsymbol{H}\boldsymbol{y} - \boldsymbol{c}^T\boldsymbol{y}\} \\ &\text{subject to} \quad -\boldsymbol{e} \le \boldsymbol{y} \le \boldsymbol{e}\ . \end{aligned} \tag{1.1}$$

Here, $\mathbf{e} \in I\!R^n$ is the vector of all ones.

A copy of the package can be obtained from the author, e-mail address hbn@imm.dtu.dk.

2. Algorithm

The algorithm for computing the solution to problem (1.1) is described in Nielsen (1996) and Madsen et al. (1997). Briefly, it can be summarized as follows.

1° Compute $\boldsymbol{R}$ as the upper triangular Cholesky factor of $\boldsymbol{H}$,

$$\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{H}\ . \tag{2.1}$$

2° Compute γ so that

$$0 < \gamma < \lambda_1\ , \tag{2.2}$$

where λ_1 is the smallest eigenvalue of $\boldsymbol{H}$.

3° Compute the upper triangular $\mathbf{A} \in I\!R^{n\times n}$ so that

$$\mathbf{A}^T\mathbf{A} = \boldsymbol{H} - \gamma\boldsymbol{I}\ . \tag{2.3}$$

4° Find the minimizer $\mathbf{x}_\gamma$ of the piecewise quadratic function

$$F(\mathbf{x}) = \frac{1}{2\gamma}\mathbf{r}^T\boldsymbol{W}\mathbf{r} + \boldsymbol{s}^T(\mathbf{r} - \frac{\gamma}{2}\boldsymbol{s}) + \frac{1}{2}\mathbf{x}^T\mathbf{x}\ , \tag{2.4a}$$

where

$$\mathbf{r} = \mathbf{r}(\mathbf{x}) = \mathbf{A}^T\mathbf{x} - \boldsymbol{c}\ , \tag{2.4b}$$

$$\boldsymbol{s} = \boldsymbol{s}(\mathbf{x}) \quad \text{with} \quad s_i(\mathbf{x}) = \begin{cases} 0 & \text{if } |r_i| < \gamma \\ \text{sign}(r_i) & \text{otherwise} \end{cases}\ , \tag{2.4c}$$

$$\boldsymbol{W} = \boldsymbol{W}(\boldsymbol{x}) = \operatorname{diag}(1 - s_i^2) \; . \tag{2.4d}$$

5° Compute the solution to (1.1) as

$$\boldsymbol{y}^* = -\frac{1}{\gamma}\boldsymbol{W}(\mathbf{x}_\gamma)\mathbf{r}(\mathbf{x}_\gamma) - \boldsymbol{s}(\mathbf{x}_\gamma) \; . \tag{2.5}$$

Step 4° is performed iteratively,

$$\boldsymbol{x} := \boldsymbol{x} + t\boldsymbol{h} \; , \tag{2.6a}$$

where $\boldsymbol{h}$ is the Newton step found by solving

$$(\boldsymbol{A}\boldsymbol{W}\boldsymbol{A}^T + \gamma\boldsymbol{I})\boldsymbol{h} = -\gamma F'(\boldsymbol{x}) \equiv -\boldsymbol{A}(\boldsymbol{W}\boldsymbol{r} + \gamma\boldsymbol{s}) - \gamma\boldsymbol{x} \; . \tag{2.6b}$$

Here, $\boldsymbol{r} = \boldsymbol{r}(\boldsymbol{x})$, $\boldsymbol{s} = \boldsymbol{s}(\boldsymbol{x})$ and $\boldsymbol{W} = \boldsymbol{W}(\boldsymbol{x})$. If $\boldsymbol{s}(\boldsymbol{x}+\boldsymbol{h}) = \boldsymbol{s}(\boldsymbol{x})$, then $t = 1$ in (2.6a) and the iteration stops. Otherwise, t is found by line search.

During computation the "dual sign vector" $\boldsymbol{s}(\boldsymbol{x})$ is given as $\boldsymbol{s}(\boldsymbol{x}) = \operatorname{round}(\boldsymbol{S}(\boldsymbol{x}))$, where

$$S_i(\boldsymbol{x}) = \begin{cases} 0 & \text{if } |r_i| < \gamma - \tau \\ \frac{1}{4}\operatorname{sign}(r_i) & \text{if } \gamma - \tau \le |r_i| < \gamma \\ \operatorname{sign}(r_i) & \text{otherwise} \end{cases} \; . \tag{2.7}$$

Here, the "fuzzyness" τ (cf. Figure 2.1 in Nielsen (1991)) is introduced to cope with effects of rounding errors in the computation of $\boldsymbol{r}(\boldsymbol{x})$. We use

$$\tau = \varepsilon_M \left(\|\boldsymbol{c}\|_\infty + \min\{n, 10\} \|\boldsymbol{A}^T\|_\infty \|\boldsymbol{x}\|_\infty \right) \; . \tag{2.8}$$

The stopping criterion $\boldsymbol{s}(\boldsymbol{x}+\boldsymbol{h}) = \boldsymbol{s}(\boldsymbol{x})$ is implemented as $\|\boldsymbol{S}(\boldsymbol{x}+\boldsymbol{h}) - \boldsymbol{S}(\boldsymbol{x})\|_\infty \le 0.75$.

In the solution of (2.6b) we use the factorization

$$\boldsymbol{L}^T\boldsymbol{L} = \boldsymbol{A}\boldsymbol{W}\boldsymbol{A}^T + \gamma\boldsymbol{I} \; , \tag{2.9}$$

where $\boldsymbol{L}$ is a lower triangular matrix. This factorization is updated during iteration, taking into account the differences between $\boldsymbol{W}(\boldsymbol{x}_{\text{old}})$ and $\boldsymbol{W}(\boldsymbol{x}_{\text{new}})$.

The problem may be given in terms of $\boldsymbol{R}$ and $\boldsymbol{c}$, in which case step 1° is omitted. Further, the algorithm is suited for "warm starts": If a problem with the same $\boldsymbol{H}$ has been solved previously, then steps 1° and 2° can be skipped. Also, step 3° can be replaced by information from the previous solution and the new $\boldsymbol{c}$. Examples are given in Section 4.

3. Use of QPBOX

A typical call of the main subroutine is

```
CALL QPBOX(N,B,LDB,C,CS,E,Y,S,W,LW,WI,LWI)
```

3.1. Parameters

`N` Integer. Size of the problem. Not changed.

`B` Double precision two-dimensional array with `LDB` rows and `N` columns. Representats the matrix $\boldsymbol{H}$, see the parameter `CS`.

`LDB` Integer. Number of elements in each column of array `B`. Must satisfy `LDB`$\ge$`N`. Is not changed.

`C` Double precision one-dimensional array with `C`$(i) = c_i$, $i = 1, \ldots,$`N`. Is not changed.

`CS` Integer. Describes how $\boldsymbol{H}$ is presented. Is not changed.

- `CS = 1` : On entry `B` should contain the symmetric, positive definite $\boldsymbol{H}$. Only the upper triangular part is used, and it is overwritten by the Cholesky factor $\boldsymbol{R}$.
- `2` : On entry `B` should contain an upper triangular matrix $\boldsymbol{R}$ so that $\boldsymbol{H} = \boldsymbol{R}^T\boldsymbol{R}$. Not changed.
- `3` : Warm start. `B` is not used, but $\boldsymbol{H} = \boldsymbol{A}^T\boldsymbol{A} + \gamma\boldsymbol{I}$, where the upper triangular $\boldsymbol{A}$ must be given in the first `N`^{2} elements of the workspace arrray `W` (e.g. left there by a previous call with the same $\boldsymbol{H}$). `S` should contain the dual sign vector, and γ should be given in `E(8)`.

`E` Double precision one-dimensional array with 14 elements, `e1`, ..., `e14`:

- `e1` = Machine accuracy ε_M. Must be set before entry. Is not changed.
- `e2` Is not used.
- `e3` = Error diagnostic:
  - `e3 = 0` : Successful run.
  - `1` : `N`<2, `N`$>$`LDB`, `CS`<1, `E(1)`≤ 0 or `E(1)`≥ 0.01.
  - `2` : Too little room in work space, cf. `LW` and `LWI`. More precise information is printed on units 6 and 7.
  - `3` : Error when computing the Cholesky factor of $\boldsymbol{H}$ (in the case `CS`$=1$).
  - `4` : $\boldsymbol{H}$ is so ill-conditioned that γ cannot be found.
  - `5` : Max no. of iterations exceeded.

e4	=	Value of $q(\boldsymbol{y})$.
e5	=	Value of gap, $F(\mathbf{x}_\gamma) + q(\boldsymbol{y}^*)$.
e6	=	Number of iterations (calls of QPSDIR).
e7	=	Number of refactorizations.
e8	=	Threshold γ.
e9	=	$\varepsilon_M \cdot \|\boldsymbol{c}\|_\infty$.
e10	=	$\varepsilon_M \cdot \min\{n, 10\} \cdot \|\boldsymbol{A}^T\|_\infty$.
e11	=	$\|\mathbf{x}\|_\infty$.
e12	=	$\|\boldsymbol{A}\|_\infty$.
e13	=	Maximum number of iterations (= N).
e14	=	Performance indicator from auxiliary routines.

Y Double precision one-dimensional array with at least N elements. Holds solution on return with e3 = 0.

S Double precision one-dimensional array with at least N+1 elements. Generalized sign vector, as defined by (2.7), augmented with S(N+1) = no. of $|S(i)| \le 0.25$, $i = 1, \ldots, N$.

W Double precision one-dimensional array with LW elements. Work space. On entry with CS = 3 the first N^2 should contain the matrix $\boldsymbol{A}$, as left there by a previous call with CS = 1 or CS = 2, and the diagnostic return e3 equal to 0 or 5.

LW Integer. Length of array W. Must satisfy $LW \ge 3N^2 + 9N + 1$. Is not changed.

WI Integer one-dimensional array with LWI elements.

LWI Integer. Length of array WI. Must satisfy $LWI \ge 2N$. Is not changed.

3.2. Auxiliary Subprograms

The package contains 21 auxiliary subprograms, which we list below together with short descriptions of their functions. In all cases parameters N, S and E are as described in Section 3.1 and W and WI are double precision and integer work space arrays, respectively. Further, we use one or more of the following parameters.

A Double precision two-dimensional array with N rows and columns. Represents the matrix $\boldsymbol{A}$, (2.3).

L Double precision two-dimensional array with N rows and columns. Represents the matrix $\boldsymbol{L}$ in the factorization (2.9).

X Double precision one-dimensional array representing the current $\boldsymbol{x}$.

R Double precision one-dimensional array representing $\mathbf{r}(\mathbf{x})$, (2.4b).

Y Double precision one-dimensional array representing the vector obtained by (2.5) with $\mathbf{x}_\gamma$ replaced by the current $\boldsymbol{x}$.

G Double precision one-dimensional array representing $F'(\mathbf{x})$, (2.6b).

H Double precision one-dimensional array representing the Newton step $\boldsymbol{h}$, (2.6).

D Double precision one-dimensional array representing $\boldsymbol{d} = \boldsymbol{A}^T\boldsymbol{h}$.

The subprograms are

```
LOGICAL FUNCTION CSTEP(N, X,R,S, H,D,E, NS, W)
```

Function: Check that the active set for $\boldsymbol{x}+\boldsymbol{h}$ is the same as for $\boldsymbol{x}$. In that case (X, R) are changed to hold $(\boldsymbol{x}+\boldsymbol{h},\ \mathbf{r}+\boldsymbol{d})$, array NS holds the corresponding sign vector, and CSTEP = true. Otherwise CSTEP = false, (X, R) are unchanged, and NS is of no use.

```
SUBROUTINE DCONST(CONST, N, X)
```

Function: Set X(i) = CONST, $i = 1, \ldots, N$.

```
SUBROUTINE DDQPL(N,L,A,LA,M,E, W)
```

Function: Downdate the factorization (2.9). Integer array LA must hold the numbers of the M columns of $\boldsymbol{A}^T$ that *leave* the dual active set.

```
SUBROUTINE DQSORT(N,X,WI)
```

Function: Quick sort of array X.

```
SUBROUTINE DQS1(I1,I2,X,SP)
```

Function: Auxiliary routine called by DQSORT.

```
SUBROUTINE ERRMSG(T)
```

Function: Print the string T of length 40 on units 6 and 7.

```
SUBROUTINE FACTOR(N,L,SO,A,S,E, W,WI)
```

Function: Update the factorization from $\boldsymbol{L}$ (entry value for L) corresponding to SO to the $\boldsymbol{L}$ (exit value for L) corresponding to S. Uses DDQPL and UDQPL.

```
SUBROUTINE GETS(N,R,E,S)
```

Function: Use (2.7) to compute the dual sign vector.

```
SUBROUTINE KINKS(KA,KK,T,K)
```

Function: Enter *kink value* in sorted array. A kink value is defined by

$$r_k + t * (\mathbf{A}^T \mathbf{h})_k = \pm\gamma \ , \qquad (3.1)$$

i.e. the kth component of $\boldsymbol{s}(\mathbf{x}+t\boldsymbol{h})$ changes value. The kink values are used in increasing order in the line search. At most the 20 smallest kink values are kept in the double precision array KA, and the corresponding k-values are kept in the integer array KK. Both arrays have length 21, and KK(21) holds the current number of kink values. Sorting by bubble sort.

```
SUBROUTINE KINKV(N,R,S,D,E,KA,KK)
```

Function: Compute all the kink values, (3.1), using KINKS to sort them.

```
DOUBLE PRECISION FUNCTION MINEST(N,BR,LDR,W)
```

Function: Returns an estimate of the smallest eigenvalue of $\boldsymbol{H} = \boldsymbol{R}^T\boldsymbol{R}$, where $\boldsymbol{R}$ is entered in the Double precision two-dimensional array BR with row dimension LDR.

```
DOUBLE PRECISION FUNCTION MINRES(P,M,R,W,WI)
```

Function: Returns the Pth smallest $|\mathtt{R}(i)|$, $i=1,\ldots,\mathtt{M}$. The value is found via quick sort, DQSORT.

```
SUBROUTINE PQFGRD(N,A,C,X,R,S,E, Y,G,W)
```

Function: G returns the gradient $F'(\mathbf{x})$, (2.6b), and Y returns $\boldsymbol{y}$ as given by (2.5) with the current $\mathbf{x}$ replacing $\mathbf{x}_\gamma$.

```
DOUBLE PRECISION FUNCTION PQFLIN(N,KA,KK,S,G,H,D,E)
```

Function: Line search. KA,KK must hold kink values as in KINKS. PQFLIN returns the value t to be used in (2.6a).

```
SUBROUTINE QPBOX1(N,BR,LDR,C,CS,E,A,L,X,Y,R,S,SO,W,WI)
```


Function: Initialization. If $\mathtt{CS} < 3$ then use QPBOX2 to compute γ and $\boldsymbol{A}$Compute starting guess $\mathbf{x}$ and corresponding $\boldsymbol{r}$, $\boldsymbol{S}$ and $\boldsymbol{y}$. The current $\boldsymbol{L}$ (2.9), is returned in L.

```
SUBROUTINE QPBOX2(N,A,C,L,E,X,R,S,SO,W,WI)
```

Function: Auxiliary program for QPBOX1.

```
SUBROUTINE QPR2A(N,BR,LDR,E, A,RNK, W)
```

Function: Given upper triangular $\boldsymbol{R}$ in the Double precision two-dimensional array BR with row dimension LDR, and the positive scalar γ in e8. Compute upper triangular $\boldsymbol{A}$ so that $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{R}^T\boldsymbol{R} - \gamma\boldsymbol{I}$. This is done by orthogonal transformation, i.e. by determining $\boldsymbol{A}$ and $\boldsymbol{Q}$ so that $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$ and

$$\begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{Q} \begin{bmatrix} \boldsymbol{A} \\ \sqrt{\gamma}\boldsymbol{I} \end{bmatrix} .$$

$\mathtt{RNK} < \mathtt{N}$ indicates that $a^2_{\mathtt{RNK}+1,\mathtt{RNK}+1}$ is not significantly positive. This implies that γ is too large. Otherwise, $\boldsymbol{A}$ is returned in A.

```
SUBROUTINE QPSDIR(N,A,L,E,G, H,D)
```

Function: Compute Newton direction $\boldsymbol{h}$ (2.6b) and $\boldsymbol{d} = \boldsymbol{A}^T\boldsymbol{h}$.

```
TRNRMS(N,A,LDA,NA1,NAI,W)
```

Function: $\mathtt{NA1} = \|\boldsymbol{A}\|_1$ and $\mathtt{NAI} = \|\boldsymbol{A}\|_\infty$, for upper triangular $\boldsymbol{A}$.

```
UCHOL(N,A,LDA,EPS, INFO)
```

Function: Cholesky factorization of symmetric, positive definite $\boldsymbol{A}$, $\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{A}$. The upper triangle of $\boldsymbol{A}$ must be stored in the Double precision two-dimensional array A with row dimension LDA, and this is overwritten by $\boldsymbol{R}$.

The program is a customized version of DPOTF2 from NETLIB, introducing the Double precision parameter EPS. The integer flag INFO has the meaning

INFO	$= 0$:	Successful run.
	< 0 :	Parameter error.
	> 0 :	The leading minor of order INFO is not pos. def.

```
SUBROUTINE UDQPL(N,L,A,EA,M, W)
```

Function: Update the factorization (2.9). Integer array EA must hold the numbers of the M columns of $\boldsymbol{A}^T$ that *enter* the dual active set.

3.3. BLAS Routines

The package uses a number of BLAS routines, cf. Dongarra et al. (1988). In this presentation we use the notation

α, β are scalars: $\alpha, \beta \in \mathbb{R}$.
$\boldsymbol{x}$, $\boldsymbol{y}$ are vectors: $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ ($\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^m$ in dgemv and dger).
$\boldsymbol{M}$ is a general matrix, $\boldsymbol{M} \in \mathbb{R}^{m \times n}$.
$\boldsymbol{T}$ is a triangular matrix, $\boldsymbol{T} \in \mathbb{R}^{n \times n}$.

We use the following routines

Name	Function
DAXPY	$\boldsymbol{y} := \alpha \boldsymbol{x} + \boldsymbol{y}$
DCOPY	$\boldsymbol{y} := \boldsymbol{x}$
DDOT	$\boldsymbol{x}^T \boldsymbol{y}$
DNRM2	$\lVert \boldsymbol{x} \rVert_2$
DSCAL	$\boldsymbol{x} := \alpha \boldsymbol{x}$
IDAMAX	$\text{argmax}\{ \lvert x_i \rvert \}$
DGEMV	$\boldsymbol{y} := \alpha \boldsymbol{M} \boldsymbol{x} + \beta \boldsymbol{y}$ or $\boldsymbol{y} := \alpha \boldsymbol{M}^T \boldsymbol{x} + \beta \boldsymbol{y}$
DGER	$\boldsymbol{M} := \boldsymbol{M} + \alpha \boldsymbol{y} \boldsymbol{x}^T$
DTRMV	$\boldsymbol{x} := \boldsymbol{T} \boldsymbol{x}$ or $\boldsymbol{x} := \boldsymbol{T}^T \boldsymbol{x}$
DTRSV	$\boldsymbol{x} := \boldsymbol{T}^{-1} \boldsymbol{x}$ or $\boldsymbol{x} := \boldsymbol{T}^{-T} \boldsymbol{x}$

4. Examples

First, consider problem(1.1) with

$$\boldsymbol{H} = \begin{bmatrix} 1.5 & -1 & 2 \\ -1 & 2.5 & -2.8 \\ 2 & -2.8 & 5.5 \end{bmatrix}, \qquad \boldsymbol{c} = \begin{bmatrix} 1.6 \\ -3.9 \\ 5.3 \end{bmatrix}.$$

The following fortran program sets up and solves this problem.

```
      PROGRAM TST1
      INTEGER          WI(6)
      DOUBLE PRECISION B(3,3),C(3),E(14),S(4),W(55),Y(3)
      DATA             C/1.6D0, -3.9D0, 5.3D0/
C
        OPEN(UNIT=7, FILE='OUT1')
        B(1,1) = 1.5D0
        B(1,2) = -1D0
        B(1,3) = 2D0
        B(2,2) = 2.5D0
        B(2,3) = -2.8D0
        B(3,3) = 5.5D0
        E(1) = 1D0 / 2D0**52
        CALL QPBOX(3,B,3,C,1,E,Y,S,W,55,WI,6)
        WRITE(7,*) 'y = ',Y(1),'  ',Y(2),'  ',Y(3)
        WRITE(7,*) 'q(y) = ',E(4)
        CLOSE(UNIT=7)
        STOP
      END
```

The solution is

$$\boldsymbol{y} = \begin{bmatrix} -0.4 \\ -1 \\ 0.6 \end{bmatrix} \quad \text{with} \quad q(\boldsymbol{y}) = -3.28 \,.$$

4.1. General Box Constraints

Next, consider the problem with general box constraints,

$$\begin{aligned} &\min_{\boldsymbol{y}} \{ q(\boldsymbol{y}) \equiv \tfrac{1}{2} \boldsymbol{y}^T \boldsymbol{H} \boldsymbol{y} - \boldsymbol{c}^T \boldsymbol{y} \} \\ &\text{subject to } \ \boldsymbol{l} \le \boldsymbol{y} \le \boldsymbol{u} \,, \end{aligned} \tag{4.1}$$

where $\boldsymbol{l}, \boldsymbol{u} \in I\!R^n$ and we assume that $\boldsymbol{l} < \boldsymbol{u}$. We can write

$$\boldsymbol{y} = \boldsymbol{a} + \boldsymbol{D}\boldsymbol{z} \ , \tag{4.2a}$$

where

$$\boldsymbol{a} = \tfrac{1}{2}(\boldsymbol{l} + \boldsymbol{u}), \qquad \boldsymbol{D} = \mathrm{diag}\big(\tfrac{1}{2}(\boldsymbol{u} - \boldsymbol{l})\big) \ . \tag{4.2b}$$

This implies $-\boldsymbol{e} \le \boldsymbol{z} \le \boldsymbol{e} \ \Leftrightarrow\ \boldsymbol{l} \le \boldsymbol{y} \le \boldsymbol{u}$, and

$$\tfrac{1}{2}\boldsymbol{y}^T\boldsymbol{H}\boldsymbol{y} - \boldsymbol{c}^T\boldsymbol{y} = \tfrac{1}{2}\boldsymbol{z}^T\boldsymbol{D}\boldsymbol{H}\boldsymbol{D}\boldsymbol{z} + \boldsymbol{z}^T\boldsymbol{D}(\boldsymbol{H}\boldsymbol{a} - \boldsymbol{c}) + \boldsymbol{a}^T(\tfrac{1}{2}\boldsymbol{H}\boldsymbol{a} - \boldsymbol{c}) \ . \tag{4.3}$$

Therefore $\boldsymbol{y}^* = \boldsymbol{a} + \boldsymbol{D}\boldsymbol{z}^*$ with $\boldsymbol{z}^*$ found as the solution to

$$\begin{aligned} &\min_{\boldsymbol{z}}\big\{\tfrac{1}{2}\boldsymbol{z}^T(\boldsymbol{D}\boldsymbol{H}\boldsymbol{D})\boldsymbol{z} - \big(\boldsymbol{D}(\boldsymbol{c} - \boldsymbol{H}\boldsymbol{a})\big)^T\boldsymbol{z}\ \big\} \\ &\text{subject to} \quad -\boldsymbol{e} \le \boldsymbol{z} \le \boldsymbol{e} \ . \end{aligned} \tag{4.4}$$

This can be found by calling `QPBOX` with $\widehat{\boldsymbol{H}} = \boldsymbol{D}\boldsymbol{H}\boldsymbol{D}$ and $\widehat{\boldsymbol{c}} = \boldsymbol{D}(\boldsymbol{c} - \boldsymbol{H}\boldsymbol{a})$.

In the special case where all lower bounds and all upper bounds are equal,

$$\boldsymbol{l} = l\boldsymbol{e}, \qquad \boldsymbol{u} = u\boldsymbol{e} \ , \tag{4.5}$$

we get

$$\boldsymbol{a} = \tfrac{1}{2}(l+u)\boldsymbol{e}, \qquad \boldsymbol{D} = \tfrac{1}{2}(u-l)\boldsymbol{I} \ ,$$

and (4.3) takes the form

$$\tfrac{1}{2}\boldsymbol{y}^T\boldsymbol{H}\boldsymbol{y} - \boldsymbol{c}^T\boldsymbol{y} = \tfrac{1}{8}(u-l)^2\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} + \tfrac{1}{2}(u-l)\boldsymbol{z}^T(\boldsymbol{H}\boldsymbol{a} - \boldsymbol{c}) + \boldsymbol{a}^T(\tfrac{1}{2}\boldsymbol{H}\boldsymbol{a} - \boldsymbol{c}) \ .$$

In this case we do not need to transform the matrix from $\boldsymbol{H}$ to $\widehat{\boldsymbol{H}}$, but find $\boldsymbol{z}^*$ as the solution of

$$\min_{\boldsymbol{z}}\big\{\tfrac{1}{2}\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} - \widetilde{\boldsymbol{c}}^T\boldsymbol{z}\ \big\} \quad \text{s. t.} \quad -\boldsymbol{e} \le \boldsymbol{z} \le \boldsymbol{e} \ , \tag{4.6a}$$

where

$$\widetilde{\boldsymbol{c}} = \frac{2}{u-l}\big(\boldsymbol{c} - \tfrac{1}{2}(l+u)\boldsymbol{H}\boldsymbol{e}\big) \ . \tag{4.6b}$$

This is exploited in the next example.

4.2. Least Squares with Constraints

Let $\boldsymbol{F} \in I\!R^{m\times n}$, $\boldsymbol{b} \in I\!R^m$ and $\boldsymbol{l}, \boldsymbol{u} \in I\!R^n$ $(m > n)$ be given. We seek $\boldsymbol{y} \in I\!R^n$ defined by

$$\min\big\{\|\boldsymbol{b} - \boldsymbol{F}\boldsymbol{y}\|_2\big\} \quad \text{s. t.} \quad \boldsymbol{l} \le \boldsymbol{y} \le \boldsymbol{u} \ . \tag{4.7}$$

By orthogonal transformation we find a matrix $\boldsymbol{Q} \in I\!R^{m\times n}$ so that $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$ and $\boldsymbol{R} = \boldsymbol{Q}^T\boldsymbol{F}$ is upper triangular. Further, let $\widetilde{\boldsymbol{b}} = \boldsymbol{Q}^T\boldsymbol{b}$. Then $\|\boldsymbol{b} - \boldsymbol{F}\boldsymbol{y}\|_2 = \|\widetilde{\boldsymbol{b}} - \boldsymbol{R}\boldsymbol{y}\|_2$, and (4.7) is equivalent with

$$\min\big\{\tfrac{1}{2}\boldsymbol{y}^T\boldsymbol{H}\boldsymbol{y} - \boldsymbol{c}^T\boldsymbol{y}\big\} \quad \text{s. t.} \quad \boldsymbol{l} \le \boldsymbol{y} \le \boldsymbol{u} \ , \tag{4.8a}$$

where

$$\boldsymbol{H} = \boldsymbol{R}^T\boldsymbol{R}, \qquad \boldsymbol{c} = \boldsymbol{R}^T\widetilde{\boldsymbol{b}} \ . \tag{4.8b}$$

Thus, we have a problem of the form (4.1) and can solve it via (4.4). Note, that `QPBOX` is called with `CS` $= 2$, since $\boldsymbol{H}$ is given by the triangular factor $\boldsymbol{R}$.

A special version of (4.7) is the problem

$$\min\big\{\|\boldsymbol{b} - \boldsymbol{F}\boldsymbol{y}\|_2\big\} \quad \text{s. t.} \quad \boldsymbol{y} \ge \boldsymbol{0} \ . \tag{4.9}$$

This is a problem with bounds of the form (4.5), where $l = 0$ and $u > 0$ is unknown. By means of (4.6) and (4.8) we see that we have to solve

$$\min_{\boldsymbol{z}}\big\{\tfrac{1}{2}\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} - \widetilde{\boldsymbol{c}}^T\boldsymbol{z}\ \big\} \quad \text{s. t.} \quad -\boldsymbol{e} \le \boldsymbol{z} \le \boldsymbol{e} \ , \tag{4.10a}$$

where

$$\widetilde{\boldsymbol{c}} = \frac{2}{u}\boldsymbol{R}^T\widetilde{\boldsymbol{b}} - \boldsymbol{R}^T\boldsymbol{R}\boldsymbol{e} \ . \tag{4.10b}$$

The value of u should be so large that none of the constraints $y_i \le u$ is active. On the other hand, it must not be so large that the term $\frac{2}{u}\boldsymbol{R}^T\widetilde{b}$ is insignificant in comparison with the other term $\boldsymbol{R}^T\boldsymbol{R}\boldsymbol{e}$ in (4.10).

A simple iterative scheme for finding u is sketched below. Note, that after the first call of `QPBOX` we use the warm start facility indicated by `CS` $= 3$.

Compute $\boldsymbol{R}$ (in `B`) and $\widetilde{\boldsymbol{b}}$; $\quad \boldsymbol{y} := \boldsymbol{R}^{-1}\widetilde{\boldsymbol{b}}$

if $\min\{y_i\} < 0$ **then**

  $\boldsymbol{g} := \boldsymbol{R}^T\boldsymbol{R}\mathbf{e}$; $\quad \boldsymbol{h} := \boldsymbol{R}^T\widetilde{\boldsymbol{b}}$; $\quad u := \|\boldsymbol{y}\|_\infty$; $\quad$ `CS` $:= 2$

  repeat

    $u := 2 * u$; $\quad \boldsymbol{c} := \frac{2}{u}\boldsymbol{h} - \boldsymbol{g}$

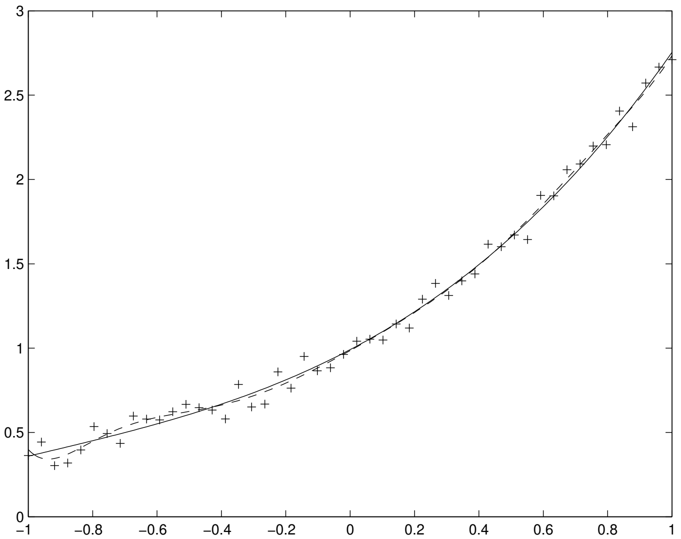
    `CALL QPBOX(N,B,LDB,C,CS,E,Z,S,W,LW,WI,LWI)`

    `CS` $:= 3$

  until $\max\{z_i\} < 1$

  $\boldsymbol{y} := \dfrac{u}{2}(\mathbf{e} + \boldsymbol{z})$

end



Data points (+), unconstrained (dotted line) and constrained fit (full line).

In the figure above we show 50 points (x_i, b_i). We have used the algorithm to find a polynomial

$$p(x) = y_1 x^9 + \cdots + y_9 x + y_{10}$$

so that $\varphi(\boldsymbol{y})^2 \equiv \sum\big(b_i - p(x_i)\big)^2$ is minimized. We get the following solution (found with the first u-value tried, $u = 8.156639$). For comparison we also

give the coefficients of the unconstrained least squares fit, and in the figure we show the corresponding curves.

i	unconstr. $\boldsymbol{y}$	constrained $\boldsymbol{y}$
1	−0.647052	0
2	2.257606	0
3	0.512288	0
4	−4.078319	0
5	0.439975	0
6	2.062174	0
7	−0.190825	0.194237
8	0.339899	0.565617
9	1.053748	1.003282
10	0.985253	0.991070
$\varphi(\boldsymbol{y})$	0.3990	0.4246

References

[1] J. Dongarra, C.B. Moler, J.R. Bunch and G.W. Stewart. 1988. *An Extended Set of Fortran Basic Linear Algebra Subprograms,* ACM Trans. Math. Soft. **14**, 1–17.

[2] K. Madsen, H.B. Nielsen and M.Ç. Pınar. 1997. *Bound Constrained Quadratic Programming Solved via Piecewise Quadratic Functions.* Report IMM-REP-1997-05, Department of Mathematical Modelling, Technical University of Denmark, DK-2800 Lyngby, Denmark. Submitted to Mathematical Programming. Available as http://www.imm.dtu.dk/~hbn/publ/TR9705.ps

[3] H.B. Nielsen. 1991. *Implementation of a Finite Algorithm for linear ℓ_1 Estimation,* Report NI-91-01, Institute for Numerical Analysis, Technical University of Denmark, DK-2800 Lyngby, Denmark.

[4] H.B. Nielsen. 1996. *Bound Constrained Quadratic Programming Solved via Piecewise Quadratic Functions. Implementation.* Report IMM-REP-1996-21, Department of Mathematical Modelling, Technical University of Denmark, DK-2800 Lyngby, Denmark. Available as http://www.imm.dtu.dk/~hbn/publ/TR9621.ps