A short description of the QPS-Reader and numerical results obtained with *qipp*

Nikolaus Rudak*

February 12, 2008

The software package *qipp* is the implentation of an interior point method for the solution of convex quadratic constrained programs. This article presents first numerical results obtained with *qipp* on the basis of the *CUTE* test library.

1 Introduction

The software package *qipp*, an object-oriented C++-package with a FORTRAN-kernel, has been developed to solve convex quadratic constrained programs. Based on the test library *CUTE* we focus on investigating the performance of *qipp*.

The formulation of a convex quadratic program for *qipp* is given in chapter 2. In [MM99] there is a standardized test library named *CUTE* containing convex quadratic programs stored in QPS format. A basic introduction of the QPS format is given in chapter 3. The data stored in the QPS format first need to be translated in a format readable for *qipp*. This translation is done by means of the QPS-Reader. A short explanation of the QPS-Reader is given in chapter 4. Finally the numerical results obtained with *qipp* by solving the programs taken from *CUTE* are presented in chapter 5.

*Institute of Applied Mathematics, TU Dortmund

2 Problem formulation for Qipp

The standard problem formulation of a convex quadratic program writes as follows:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \ & \tfrac{1}{2} x^T Q x + c^T x \\ s.t. \ & \underline{c} \leq Cx \leq \overline{c} \\ & Ax = b \\ & \underline{x} \leq x \leq \overline{x}, \end{aligned} \tag{QP}$$

where $Q \in \mathbb{R}^{n_x \times n_x}$, $n_x \in \mathbb{N}$, represents the quadratic and $c \in \mathbb{R}^{n_x}$ the linear part of the objective function. Q is symmetric and positive semi-definite. The matrix $C \in \mathbb{R}^{m_y \times n_x}$, $m_y \in \mathbb{N}$, and the vectors $\underline{c}, \overline{c} \in \mathbb{R}^{m_y}$ are part of the inequality constraints, the matrix $A \in \mathbb{R}^{m_z \times n_x}$, $m_z \in \mathbb{N}$, and the vector $b \in \mathbb{R}^{m_z}$ stand for the equality constraints. The vectors $\underline{x} \in \mathbb{R}^{n_x}$ and $\overline{x} \in \mathbb{R}^{n_x}$ represent upper and lower bounds, respectively, of the box constraints.

3 The QPS-Format

In Linear Programming there is a usual way of presenting LP problems: the so called MPS format developed by IBM. Further explanations of the MPS format are given in [MK81]. As an extension of the MPS format the QPS format allows to store the QP described in section 2. Maros and Meszaros developed the QPS format and described it in [MM99]. In this section we give a short description of the QPS format. In the following the term "qps-file" means a file containing a QP stored in QPS format.

A qps-file consists of the following eight sections `NAME`, `ROWS`, `COLUMNS`, `RHS`, `BOUNDS`, `RANGES`, `QUADOBJ` and `ENDATA` arranged among each other. The rows of a qps-file consist of six fields in each section. Fields start in column 1, 5, 15, 25, 40, 50 and contain the data of the QP. Exceptions are the rows containing the name of each section which start in column 0.

Some basic descriptions about the different sections of a qps-file:

- `NAME`: labels the name of the QP,
- `ROWS`: this section specifies each row of the constraint matrix labels and the row type. The order is not important. There are different row types:
  - `N`: stands for objective function ,
  - `L`: stands for lower inequality constraints,
  - `G`: stands for upper inequality constraints,
  - `E`: stands for equality constraints,
- `COLUMNS`: this section defines the name of the variables, the coefficients of the objective function and all the nonzero elements of the constraint matrix. Thus the data for the matrices C, A and the vector c is stored in this section,

- RHS und RANGES: these sections contain the values of the vectors $\underline{c}$, b and $\overline{c}$. Components of $\underline{c}$ not regarded in RHS will be treated as zero,
- BOUNDS: this section defines the bounds of variables and contains the values for $\underline{x}$ and $\overline{x}$. There are different bound types:
  - LO: lower bound,
  - UP: upper bound,
  - FR: no bounds ,
  - FX: lower bound is equal to upper bound,

  Components of $\underline{x}$ not regarded will be treated as zero,
- QUADOBJ: this section defines the coefficients for the quadratic part of the objective function and contains the entries of the lower triangular part of the symmetric matrix Q. These elements appear in the same order as in section COLUMNS.

In the following we explain the QPS format based on an example.

Example 1: Let us consider the qps-file problem.qps:

```
NAME          problem
ROWS
 N  obj
 G  c1
COLUMNS
    a         obj                  1   c1                   1
    b         obj                  1   c1                   1
RHS
    rhs       c1                  10
BOUNDS
RANGES
QUADOBJ
    a         a                    1
    a         b                    2
    b         b                    7
ENDATA
```

Obviously this program consists of two rows. The objective function defined by type N is named obj. Because of the type G the inequality constraint c1 is bounded above. In section COLUMNS we can find the entries for the linear part c of the objective function and the Matrix C. In the first row of the section COLUMNS we find the value 1 as coefficient of the variable a in objective function obj and in the inequality constraint c1. The second row contains the value 1 as coefficient for the variable b in the objective function obj as well as in inequality constraint c1. In the next section RHS we find the value 10 for the upper bound of c1. Because the section BOUNDS is empty we define the lower bounds for the variables a, b as zero. The next section QUADOBJ contains the entries for the matrix Q. Those entries are the coefficients of the nonlinear part of the

objective function. As we can see, the coefficient of a^2 is 1 and the coefficients of ab und ba are 2. The coefficient of b^2 is 7. Now the quadratic program from the qps-file above can be written as follows:

$$\min_{a,b\in\mathbb{R}} \ \frac{1}{2}\begin{pmatrix} a & b \end{pmatrix}\begin{pmatrix} 1 & 2 \\ 2 & 7 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} + \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix}$$
$$s.t. \quad \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} \geq 10$$
$$\begin{pmatrix} a \\ b \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

As already mentioned above the quadratic programs from the standardized test library *CUTE* are stored in QPS format. By means of the QPS-Reader we prepare the data of these programs for *qipp*. Our main interest is to verify whether *qipp* manages to solve the programs in *CUTE*. Notice that these programs are large scaled or ill conditioned and thus are widely used as test problems for QP solvers.

In the next section some explanations of the QPS-Reader are given.

4 The QPS-Reader

The QPS-Reader converts the data from a qps-file into the format *qipp* requires. After this conversion the QPS-Reader calls *qipp* in order to solve the program. In the following we give some explanation on how to call the QPS-Reader.

Below `problem.qps` from chapter 3 is used as examplary qps-file. To call the QPS-Reader in order to solve the program the following command has to be entered:

```
qpsreader.exe problem.qps
```

Now the QPS-Reader starts to read the qps-file and converts the data. After converting the data the QPS-Reader starts *qipp* and the results are presented in the following form:

```
#  Iteration-Table:
# -----------------
# __It  ___resnorm  ___dualgap  ______mu  ___sigma  ___alpha  _____dnorm
#    0      3.1623     -31.623         0         1         1          10
#    1      1005.6      -10087  1.02e+06         1         1          10
#    2  1.1369e-13  3.2168e+05  1.07e+05   0.00328         1          10
#    3  1.1369e-13       47134  1.57e+04    0.0156         1          10
#    4  5.6843e-14      6896.6   2.3e+03    0.0156         1          10
#    5           0       998.3       333    0.0154         1          10
#    6  1.2434e-14      131.36      43.8    0.0135         1          10
#    7  5.3291e-15      7.3958      2.47   0.00367     0.986          10
#    8  1.7764e-15    0.065449    0.0218  0.000129     0.998          10
#    9  3.5527e-15  8.5862e-07  2.86e-07  3.22e-08         1          10
#   10  1.7764e-15 -2.8422e-14   5.7e-21  7.82e-23         1          10

*** SUCCESSFUL TERMINATION ***
  -> solution: x=[10, 1.7073e-21];
  -> objective value: 60.00000
```

This table consists of some control paramaters and at the end there is the solution and the objective value. Here some explanation on the control paramaters:

- `It`: number of iterations,
- `resnorm`: norm of the residual right hand side vector of the KKT system,
- `dualgap`: value for the duality gap,
- `mu`: duality measure,
- `sigma`: a centering parameter,
- `alpha`: steplength,
- `dnorm`: *qipp* calculates the norm of all data and returns the biggest value as dnorm.

5 Numerical results

We tested the performance of *qipp* on the basis of the testlibrary *CUTE*. In this section we present the numerical results obtained with *qipp*. The numbers n_x, m_y, m_z denote the problem size as explained in section 2. The column K shows the number of iterations, and μ denotes the duality gap. The column $f_{\mathrm{Q}}(x^*)$ displays the optimal objective function value obtained with *qipp*. As a comparison, the column $f_{\mathrm{B}}(x^*)$ shows the optimal objective function value obtained with BPMPD. The interior point solver BPMPD developed by Mészáros makes use of sparsity, data preprocessing, etc. [Més99]. Thus it may well serve as a good comparison where, however, *qipp* does not possess these features in its current version. The results obtained with BPMPD can be found in [MM99].

5.1 *CUTE*

In the following we present the results obtained with *qipp* on programs taken from the *CUTE*-library.

NAME	n_x	m_y	m_z	K	μ	$f_{\mathrm{Q}}(x^*)$	$f_{\mathrm{B}}(x^*)$
aug2dc	20200	10000	0	19	3.72529e-09	6.49807e+06	1.81837e+06
aug2dqp	20200	10000	0	26	-3.33165e-08	6.23701e+06	6.23701e+06
aug2d	20200	10000	0	19	-8.00937e-08	6.23701e+06	1.68741e+06
aug2dcqp	20200	10000	0	22	1.30824e-08	6.49813e+06	6.49813e+06
aug3dc	3873	1000	0	19	2.72848e-12	959.825	771.262
aug3dcqp	3873	1000	0	22	3.54716e-09	993.362	993.362
aug3dqp	3873	1000	0	24	1.31976e-08	675.238	675.238
aug3d	3873	1000	0	18	1.13687e-12	675.037	554.068
cvxqp1_m	1000	500	0	14	-7.34871e-10	1.08751e+06	1.08751e+06

cvxqp1_s	100	50	0	10	-3.75723e-12	11590.7	11590.7
cvxqp1_l	10000	5000	0	48	1.25179e+16	2.15645e+09	1.08705e+08
cvxqp2_m	1000	250	0	12	-9.24047e-10	820155	820155
cvxqp2_s	100	25	0	11	2.72647e-10	8120.94	8120.94
cvxqp2_l	10000	2500	0	18	4.30282e-07	8.18425e+07	8.18425e+07
cvxqp3_m	1000	750	0	47	4.77184e+12	1.34112e+07	1.36283e+06
cvxqp3_s	100	75	0	11	8.64041e-10	11943.4	11943.4
cvxqp3_l	10000	7500	0	16	2.97829e-06	1.15711e+08	1.15711e+08
dtoc3	14999	9998	0	11	7.85252e+06	3.66137e+06	235.262
dual1	85	1	0	14	5.65115e-11	0.035013	0.035013
dual2	96	1	0	11	6.16116e-11	0.0337337	0.0337337
dual3	111	1	0	10	1.32363e-09	0.135756	0.135756
dual4	75	1	0	12	9.47459e-11	0.746091	0.746091
dualc1	9	1	214	18	1.20702e-20	6259.82	6155.25
dualc2	7	1	228	15	9.09495e-13	3577.1	3551.31
dualc5	8	1	277	13	3.75594e-12	427.232	427.232
dualc8	8	1	502	15	1.22453e-09	22508.3	18309.4
genhs28	10	8	0	10	2.22045e-16	0.928915	0.927174
gouldqp2	699	349	0	14	2.52548e-09	0.000184275	0.000184275
gouldqp3	699	349	0	11	5.23381e-09	2.06278	2.06278
hs21	2	0	1	10	1.38778e-17	-99.96	-99.96
hs35	3	0	1	11	6.66134e-16	0.111111	0.111111
hs51	5	3	0	10	-2.47864e-16	0	8.88178e-16
hs52	5	3	0	5	-2.66021e-21	6	5.32665
hs53	5	3	0	6	-1.66533e-16	4.09302	4.09302
hs76	4	0	3	12	4.44089e-16	-4.68182	-4.68182
hs118	15	0	17	15	-1.80667e-14	664.82	664.82
hs268	5	0	5	12	2.82654e-11	7.09129	5.73107e-07
hs35mod	3	0	1	21	4.36557e-11	0.25	0.25
hues-mod	10000	2	0	18	1.63913e-07	3.48245e+07	3.48247e+07
huestis	10000	2	0	20	-0.00170898	3.48245e+11	3.48247e+11
ksip	20	0	1001	26	1.13746e-10	0.59445	0.575798
liswet1	10002	0	10000	18	-5.37434e-12	36.1224	36.1224
liswet2	10002	0	10000	22	1.49474e-08	24.9981	24.9981
liswet3	10002	0	10000	31	1.28739e-08	25.0013	25.0012
liswet4	10002	0	10000	33	9.07514e-08	25.0001	25.0001
liswet5	10002	0	10000	27	5.74806e-08	25.0343	25.0343
liswet6	10002	0	10000	29	1.89023e-07	24.9957	24.9957
liswet7	10002	0	10000	18	-2.42642e-10	498.841	498.841
liswet8	10002	0	10000	30	6.80841e-10	1677.92	7144.7
liswet9	10002	0	10000	28	1.30644e-10	2101.65	1963.25
liswet10	10002	0	10000	30	1.5224e-09	1287.02	49.4858
liswet11	10002	0	10000	25	3.59851e-09	1286.69	49.524
liswet12	10002	0	10000	30	2.74666e-10	1905.98	1736.93

lotschd	12	7	0	11	9.09495e-13	2398.42	2398.42
mosarqp1	2500	0	700	19	2.24617e-10	-952.875	-952.875
mosarqp2	900	0	600	18	1.30171e-11	-1597.48	-1597.48
powell20	10000	0	10000	49	-5.15824e+10	1.86095e+11	5.20896e+10
primal1	325	0	85	22	2.2262e-09	-0.0342801	-0.035013
primal2	649	0	96	22	4.9938e-09	-0.0349932	-0.0337337
primal3	745	0	111	21	7.15259e-09	-0.142833	-0.135756
primal4	1489	0	75	21	1.44754e-08	-0.723685	-0.746091
primalc1	230	0	9	31	3.36513e-11	-4983.32	-6155.25
primalc2	231	0	7	20	0	-2.77255	-3551.31
primalc5	287	0	8	14	2.84217e-13	-461.115	-427.232
primalc8	520	0	8	24	1.93268e-12	-914.595	-18309.4
qpcblend	83	43	31	20	1.53932e-15	1.53874e-15	-0.00784254
qpcboei1	384	9	342	11	-1.47615e+12	1.24185e+07	1.15039e+07
qpcboei2	143	4	162	12	-4.57233e+14	1.0375e+07	8.17196e+06
qpcstair	467	209	147	17	-7.75686e+08	1.85655e+07	6.20439e+06
s268	5	0	5	12	2.82654e-11	7.09129	5.73107e-07
stcqp1	4097	2052	0	13	-2.29647e-11	155144	155144
stcqp2	4097	2052	0	13	1.5552e-10	22327.3	22327.3
tame	2	1	0	3	9.86076e-32	4.93038e-32	0
ubh1	18009	12000	0	11	-1.05468e+11	72.6932	1.116
yao	2002	0	2000	21	-4.83138e-13	197.704	197.704
zecevic2	2	0	2	6	4.10029e-13	-4.125	-4.125

5.2 Conclusion

Most of the problems are successfully solved by *qipp*. Some results, however, differ from the results obtained with BPMPD. *Qipp* was not able to solve the programs `ubh1`, `qpcstair`, `dtoc3`, `cvxqp1_l`, `cvxqp3_m`, `powell20`, `qpcboei1` and `qpcboei2`. Let us have a closer look at these problems.

The program `cvxqp1_l` is large and badly scaled and possesses 10000 variables. The region of feasible points is relatively small. The problem `cvxqp3_m` has a similar structure as `cvxqp1_l` and also has a relatively small feasible region. Hence *qipp* might struggle with ill conditioning. In both programs *qipp* terminates after 48 and 47 iterations with a large duality gap. The program `dtoc3` is also a large scaled program with 14999 variables. There are no box constraints and the quadratic matrix Q is a diagonal matrix with small values. There are only equality constraints, and two variables get a fixed value at the beginning. Here *qipp* stops after a few iterations and defines the program as infeasible. As in `dtoc3` the programs `qpcstair` and `ubh1` have fixed variables. In the case of fixed variables it seems that *qipp* has difficulties to calculate a proper search direction. The programs `qpcboei1` and `qpcboei2` possess a diagonal matrix Q and a scattered matrix C. The lower and upper bound vectors of the inequality constraints are similar. Therefore the feasible region is relatively small and *qipp* stops

after a few iterations declaring the programs as infeasible. In the program `powell20` Q is the identity matrix and only inequality constraints are defined. The matrix C is also diagonal with alternating entries -1 and 1. *Qipp* terminates after 49 iterations with a large duality gap.

However, in our first test *Qipp* solved most of the programs succesfully. In view of improving *qipp* our future challenge is to implement additional features so that *qipp* is able to solve also the problematic programs mentioned above.

References

[MK81] Bruce A. Murtagh and Peter Klaus. *Advanced Linear Programming.* McGrawhill, 1981.

[MM99] Istvan Maros and Csaba Mészáros. A repository of convex quadratic programming problems. *Optimization Methods and Software*, 11-12, 1999.

[Més99] Csaba Mészáros. The bpmpd interior point solver fo convex quadratic programs. *Optimization Methods and Software*, 11-12, 1999.