BPMPD user's manual
Version 2.20

Csaba Mészáros *

Imperial College, Deparment of Computing

Abstract

The purpose of this document is to describe a software package, called BPMPD, which implements the infeasible primal–dual interior point method for linear and quadratic programming problems. This manual describes how to prepare data to solve with the package, how to use BPMPD as a callable solver library and which algorithmic options can be specified by the user.

1 Problem formulation

BPMPD is a software package to solve linear (LP) and convex quadratic (QP) problems. For simplicity we will introduce here the quadratic programming problem which includes the linear programming as special case. Without loss of generality, the convex QP problem is usually assumed to be in the following form:

$$\begin{aligned} \min \quad & c^T x + \tfrac{1}{2} x^T Q x, \\ \text{subject to} \quad & Ax = b, \\ & x \geq 0 \end{aligned} \tag{1}$$

where $A \in R^{m \times n}$ of full row rank, $Q \in R^{n \times n}$ symmetric positive semidefinite and $c, x \in R^n$, $b \in R^m$. It is to be noted that an $1/2$ explicit term is given in the quadratic matrix. The dual associated with this problem can be written as

$$\begin{aligned} \max \quad & b^T y - \tfrac{1}{2} x^T Q x, \\ \text{subject to} \quad & A^T y + z - Qx = c, \\ & z \geq 0 \end{aligned} \tag{2}$$

*The author was supported by EPSRC grant No. GR/J52655

where $z \in R^n$ and $y \in R^m$. The constrains of the above problems define convex polyhedrons which are nonempty if a feasible primal–dual solution exists. The Karush–Kuhn–Tucker optimality conditions for (1) and (2) are:

$$\begin{aligned} Ax &= b, && (3)\\ A^T y + z - Qx &= c, && (4)\\ Xz &= 0, && (5)\\ (x, z) &\geq 0 && (6) \end{aligned}$$

where $X = diag(x_1, ..., x_n)$.

BPMPD implements the infeasible primal–dual interior point algorithm [10, 1] that is BPMPD generates a sequence of iterates

$$(x^k, y^k, z^k) \qquad k = 0, 1, 2, ...$$

which satisfies a strict positivity condition $(x^k, z^k) > 0$, but feasibility (3,4) and optimality (5) reached as $k \longrightarrow \infty$. The primal–dual interior point algorithm is based on the application of the Newton method to solve the perturbed KKT system. The perturbation is introduced to retain positivity in (x, z) by replacing (5) with

$$Xz = \mu e$$

where μ is the nonnegative centering parameter and e is the n vector of ones.

To derive the primal-dual algorithm one should:

- replace nonnegativity constraints on the variables with logarithmic barrier penalty terms;
- move equality constraints to the objective with the Lagrange transformation to obtain an unconstrained optimization problem and write first order optimality conditions for it; and
- apply Newton's method to solve these first order optimality conditions (i.e. to solve a system of nonlinear equations).

Replacing nonnegativity constraints with the logarithmic penalty terms gives the following logarithmic barrier problem:

$$\begin{aligned} \text{minimize} \quad & c^T x + \tfrac{1}{2} x^T Q x - \mu \sum_{j=1}^{n} \ln x_j \\ \text{subject to} \quad & Ax = b. \end{aligned} \qquad (7)$$

The Lagrangian for (7) is

$$L(x,s,y,w,\mu) = c^T x + \frac{1}{2}x^T Q x - \mu \sum_{j=1}^{n} \ln x_j - y^T(Ax-b)$$

and the first order optimality conditions for (7) are:

$$\begin{array}{rcl} \nabla_x L = & c - \mu X^{-1} e - A^T y + Qx & = 0, \\ \nabla_y L = & b - Ax & = 0. \end{array}$$

By introducing $Z = \mu X^{-1}$ we can write the first order optimality conditions of (7) as

$$\begin{array}{rcl} Ax & = & b, \\ A^T y + z - Qx & = & c, \\ XZe & = & \mu e. \end{array} \tag{8}$$

Let us observe that the first two of the above equations are linear and force primal and dual feasibility of the solution. The last equation is nonlinear and depends on the barrier parameter μ. It becomes the complementarity condition for $\mu = 0$, which together with the feasibility constraints provides optimality of the solutions.

It can be seen that (8) is identical to the Karush-Kuhn-Tucker (KKT) system for the LP problem, in which the complementarity conditions are perturbed by μ. Hence, (8) is called the perturbed KKT conditions.

A nonnegative solution of (8) is called an *analytic center*. It clearly depends on the value of the barrier parameter μ. The set of such solutions $(x(\mu))$ and $(y(\mu), z(\mu))$ defines a trajectory of centers for the primal and dual problem, respectively and is called the *central path*. The quantity

$$g = x^T z + s^T w,$$

measures the error in the complementarity and is called a *complementarity gap*. Note that for a feasible point, this value reduces to the usual duality gap. For a μ-center, for example,

$$g = 2\mu e^T e = 2n\mu, \tag{9}$$

and it vanishes at an optimal solution.

One iteration of the primal-dual algorithm makes one step of Newton's method applied to the first order optimality conditions (8) with a given μ and then μ is

updated (usually decreased). The algorithm terminates when the infeasibility and the complementarity gap are reduced below predetermined tolerances.

Given an $x, z \in \mathcal{R}^n_+, \quad y \in \mathcal{R}^m$, Newton's direction is obtained by solving the following system of linear equations

$$\begin{bmatrix} A & 0 & 0 \\ Q & A^T & I \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} \xi_b \\ \xi_c \\ \mu e - XZe \end{bmatrix}, \tag{10}$$

where

$$\begin{aligned} & \xi_b &=& b - Ax, \\ \text{and} \qquad & \xi_c &=& c - A^T y - z + Qx \end{aligned}$$

denote the violations of the primal and the dual constraints, respectively. We call the linear system (10) the Newton equations system.

Note that the primal-dual method does not require feasibility of the solutions (ξ_b, and ξ_c might be nonzero) during the optimization process. Feasibility is attained during the process as optimality is approached. It is easy to verify that if a step of length one is made in the Newton's direction (10), then feasibility is reached immediately. This is seldom the case as a smaller stepsize usually has to be chosen (a damped Newton iteration is taken) to preserve positivity of x and z. If this is the case and a stepsize $\alpha < 1$ is applied, then infeasibilities ξ_b, and ξ_c are reduced by a factor $(1 - \alpha)$.

Once the system (10) has been solved, the maximum stepsizes in primal space (α_P) and dual space (α_D) are computed such that the nonnegativity of variables is preserved:

$$\begin{aligned} \alpha_P &= \frac{1}{\max\limits_{k=1\ldots n} \left\{1, -\frac{\Delta x_k}{x_k}\right\}}, \\ \alpha_D &= \frac{1}{\max\limits_{k=1\ldots n} \left\{1, -\frac{\Delta z_k}{z_k}\right\}}. \end{aligned}$$

Then the common steplengt is defined as $\alpha = \min(\alpha_P, \alpha_D)$. These stepsizes are slightly reduced with a factor $\alpha_0 < 1$ to prevent hitting the boundary. Finally a new iterate is computed as follows

$$\begin{aligned} x &\leftarrow x + \alpha_0 \alpha \Delta x, \\ y &\leftarrow y + \alpha_0 \alpha \Delta y, \\ z &\leftarrow z + \alpha_0 \alpha \Delta z. \end{aligned} \tag{11}$$

After making the step, the barrier parameter μ is decreased by a given factor and the process is repeated.

Interior point algorithms terminate when the first order optimality conditions (8) are satisfied with some predetermined tolerance. In the case of the primal–dual method, this translates to the following conditions imposed on the relative primal and dual feasibility and the relative duality gap

$$\frac{||Ax - b||}{1 + ||b||} \leq 10^{-p},$$

$$\frac{||A^T y + z - w - c||}{1 + ||c + Qx||} \leq 10^{-p},$$

$$\frac{|c^T x + \frac{1}{2} x^T Q x - (b^T y - \frac{1}{2} x^T Q x)|}{1 + |b^T y + \frac{1}{2} x^T Q x|} \leq 10^{-p}$$

where p is the number of digits accurate in the solution. An 8–digits exact solution ($p = 8$) is typically required in the literature.

In each iteration BPMPD uses Mehrotra's predictor corrector technique [5] to compute the search direction and the centering parameter μ. Furthermore, the multiple correction technique [3] is also applied.

2 Installing BPMPD

BPMPD is entirely written in Fortran 77 and is believed to be portable. Only the timing routine has to be adjusted to the actual platform and compiler. In the distribution this timing routine is provided for a number of platforms and compilers and the relevant one can be easily activated. The C version of the timing routine is also included which is standard under UNIX.

BPMPD can be compiled and linked with a Fortran 77 compiler system. Alternatively, the Fortran 77 source code can be translated to C by using the public domain *f2c* system.

To install the executable version of BPMPD one parameter, *realmxx* has to be set in `bpmain.f` for memory allocation prior to compile the source file. The value of this parameter depends on the available memory of the platform and determines the memory allocation. BPMPD will allocate about $14 \cdot$*realmmx* bytes of memory, that is for a computer platform with 64 megabyte operative memory *realmmx* can be set to about 4500000.

To compile the callable library form of BPMPD, the source files `bpmain.f`, `minput.f`, `mpsinp.f`, `mpsout.f` and `convert.f` have to be omitted. An interface, which simplifies the use of BPMPD as a subroutine can be created (we will describe such an interface in one of the forthcoming sections) and included to the source files. We would like to note that only one routine, *mprnt* (located in the file `mprnt.f`) contains write statements to external files.

3 Solution of a problem

The first step of the solution process is to generate an input file which describes the optimization problem. BPMPD supports the industry–standard MPS format with some extensions. In the input file A, b and c, furthermore the bounds on individual variables and rows are specified by the usual ROWS, COLUMNS, RHS, BOUNDS, RANGES records of the MPS format [11]. To specify the quadratic part of the problem, a new section "QUADOBJ" is introduced. It serves to specify the lower triangular part of the quadratic objective in that relative column order as it has been given in the COLUMNS section. Otherwise the data records in the QUADOBJ section are of the same structure than that of the COLUMNS section.

As example for the problem formulation, let us consider the following quadratic programming problem:

$$\begin{aligned} \min\ f(x,y) = \ & 4 + 1.5x - 2y + 4x^2 + 2xy + 5y^2, \\ & 2x + y \geq 2, \\ & -x + 2y \leq 6, \\ & 0 \leq x \leq 20,\ y \geq 0. \end{aligned}$$

After implied rewriting:

$$\begin{aligned} \min\ f(x,y) = \ & 4 + 1.5x - 2y + \tfrac{1}{2}(8x^2 + 2xy + 2yx + 10y^2), \\ & 2x + y \geq 2, \\ & -x + 2y \leq 6, \\ & 0 \leq x \leq 20,\ y \geq 0. \end{aligned}$$

Thus the Q matrix is

$$\begin{bmatrix} 8 & 2 \\ 2 & 10 \end{bmatrix}$$

for which the lower triangular part (to be given) is:

$$\begin{bmatrix} 8 & \\ 2 & 10 \end{bmatrix}.$$

Below we have shown the MPS file corresponding to the example. Note that the additive term in the objective function is given in the RHS section by opposite sign.

```
NAME          QP example
ROWS
 N  obj
 G  r1
 L  r2
COLUMNS
    c1        r1                 2.0   r2                -1.0
    c1        obj                1.5
    c2        r1                 1.0   r2                 2.0
    c2        obj               -2.0
RHS
    rhs1      obj               -4.0
    rhs1      r1                 2.0   r2                 6.0
BOUNDS
 UP bnd1      c1                20.0
QUADOBJ
    c1        c1                 8.0
    c1        c2                 2.0
    c2        c2                10.0
ENDATA
```

After the input file has been created the problem can be solved. The optimizer can be called by the *bpmpd* command. First, BPMPD will read its parameter file called *bpmpd.par* if presented. If the parameter file exists and the problem file name is specified in it, BPMPD will start to read the problem and to solve it. Otherwise BPMPD will prompt the user for the the input file name. It is assumed by BPMPD that the extension of the input file is either *.mps* or *.qps*. The name of the MPS input file has to be given by the user without extension.

Depending on the specifications BPMPD will create an optimization report file (with extension *.log*) and a solution file (with extension *.out*).

4 Using the package as callable library

BPMPD is a modularized software package, written in Fortran 77, in which the problem input/output and optimization are completely separated. As a stand-alone solver, BPMPD is prepared to read optimization problems from MPS files. The main program (`bpmain.f`) contains subroutine call to the MPS input reader (*finput*), to the optimizer driver routine (*solver*) and to the output writer routine (*mpsout*). Additionally, the "built–in" values of the optimization parameters are defined here.

BPMPD operates in a column file and in a row file. The column file consists of two working arrays (one double precision called *colnzs* and one integer called *colidx*) which are of the same dimension. The row file is one integer working array (called *rowidx*). The column file contains the A and the lower off–diagonal part of Q as part of the input in form of sparse vectors. They need to be placed continuously from the first position of the working arrays. The remaining part of the column file will be used to hold the factorization during the algorithm, while the purpose of the row file is to support row-wise representation of data when necessary.

The problem and memory dimensions are put in the common block /dims/ which has the following structure:

```
common/dims/ n,qn,n1,m,mn,nz,qnz,cfree,pivotn,denwin,rfree
integer*4    n,qn,n1,m,mn,nz,qnz,cfree,pivotn,denwin,rfree
```

The following values needed to be set prior to calling the optimization driver routine:

Name	Description
n	Number of variables.
qn	Number of quadratic variables.
n1	= n+1.
m	Number of rows in A.
mn	= n+m.
nz	Number of nonzeros in A.
qnz	Number of nonzeros in Q.
cfree	Lenght of the column file.
rfree	Length of the row file.

The optimization driver routine can serve as a callable library. The QP problem has to be passed to this subroutine as input. BPMPD assumes the problem in the

following form:

$$\begin{aligned} \min \quad & c^T x + \tfrac{1}{2} x^T Q x, \\ \text{subject to} \quad & Ax - Is = b, \\ & u \geq (x, s)^T \geq l \end{aligned}$$

where $s \in R^m$ are slack variables for rows and $u, l \in R^{n+m}$ are (possibly infinite) upper and lower bouns. BPMPD computes the primal–dual solution corresponding this formulation.

The input parameters for this subroutine are:

1.

Name	Type	Length	Description
obj	double prec.	n	Linear part of the objective function.
rhs	double prec.	m	Right hand side.
ubound	double prec.	n+m	Upper bound of variables.
lbound	double prec.	n+m	Lower bounds of variables.
colpnt	integer	n+1	Column pointers of A.
colidx	integer	cfree	Indices of the columns in A and Q.
colnzs	double prec.	cfree	Nonzero values in A and Q.
qdiag	double prec.	n	Diagonal of the quadratic objective.
qpnt	integer	n+1	Column pointers of Q.
big	double prec.	1	Represents infinity (10^{30}).
addobj	double prec.	1	Additive term to the objective.

Note if *qnz=0* then *qpnt,* if *qn=0* then both *qpnt* and *qdiag* can be dummy parameter. The subroutine results in the following outgoing parameters:

Name	Type	Length	Description
xs	double prec.	n+m	Last (optimal) primal iterate.
dv	double prec.	m	Last (optimal) dual iterate.
dspr	double prec	n+m	Last (optimal) dual slack iterate.
code	integer	1	Termination code.
iter	integer	1	Number of iterations during optimization.
opt	double prec.	1	Final primal objective function value.

The termination code can have one of the following values:

Code value	Statement
< 0	Not enougt memory, optimization was not performed.
1	Suboptimal solution.
2	Optimal solution.
3	Problem is dual infeasible.
4	Problem is primal infeasible.

As an example the following code presents an interface to the optimization driver routine:

```
      subroutine bpmpd(mm,nn,qnn,qnnz,isiz,rsiz,colnzs,colidx,
     x colpnt,qpnt,ubound,lbound,obj,qdiag,rhs,addobj,xs,dv,
     x dspr,vstat,opt,big,code)

      integer*4 mm,nn,qnn,qnnz,isiz,rsiz,colidx(isiz),
     x colpnt(nn+1),qpnt(nn+1),vstat(mm+nn),code
      real*8 colnzs(rsiz),ubound(nn+mm),lbound(nn+mm),
     x obj(nn),qdiag(nn),rhs(mm),addobj,xs(mm+nn),dv(mm),
     x dspr(mm+nn),opt,big
c
      common/dims/ n,qn,n1,m,mn,nz,qnz,cfree,pivotn,denwin,rfree
      integer*4    n,qn,n1,m,mn,nz,qnz,cfree,pivotn,denwin,rfree

      common/logprt/ loglog,lfile
      integer*4      loglog,lfile
c
c Local variables to read parameter file
c
      real*8 bigbou,at
      character*8 objnam,rhsnam,bndnam,rngnam
      character*99 namstr,outstr,ofnam
      integer*4 i,j,k,l,iter,fnzmax,fnzmin
c
c Setting up problem dimensions
c
```

```
      qn=qnn
      qnz=qnnz
      n=nn
      m=mm
      n1=n+1
      mn=m+n
      nz=colpnt(n+1)-1
c
c Computing the length of the row and column files
c
      cfree=rsiz-(13*mn+3*m+3)
      rfree=isiz-(11*mn+3)-cfree
      if(rfree.lt.0)then
        code=-2
        goto 999
      endif
c
c Set printing log to "stdout only", we do not open logfile
c
      loglog=1
c
c Read-in optimization parameters.
c
      outstr='bpmpd.par'
      call readpar(outstr,i,j,k,namstr,ofnam,
     x objnam,rhsnam,bndnam,rngnam,bigbou,at,l)
c
c Call the optimizer
c
      code=0
      call solver(
     x obj,rhs,lbound,ubound,colnzs(rsiz-mn),colnzs(rsiz-2*mn),xs,
     x colnzs(rsiz- 3*mn),colnzs(rsiz- 4*mn),colnzs(rsiz- 5*mn),dspr,
     x colnzs(rsiz- 6*mn),colnzs(rsiz- 7*mn),colnzs(rsiz- 8*mn),
     x colnzs(rsiz- 9*mn),colnzs(rsiz-10*mn),dv,
     x colnzs(rsiz-10*mn-  m),colnzs(rsiz-10*mn-2*m),
     x colnzs(rsiz-10*mn-3*m),colnzs(rsiz-11*mn-3*m),
```

```
     x colnzs(rsiz-12*mn-3*m),colnzs(rsiz-13*mn-3*m),
     x qdiag,colnzs,
     x colidx(isiz-    n),colidx(isiz-mn),colpnt,colidx(isiz-2*mn),
     x colidx(isiz- 3*mn),colidx(isiz-4*mn),vstat,colidx(isiz-5*mn),
     x colidx(isiz- 6*mn),colidx(isiz -7*mn),colidx(isiz- 8*mn),
     x colidx(isiz- 9*mn),colidx(isiz-10*mn),qpnt,
     x colidx(isiz-11*mn),colidx,colidx(cfree+2),
     x code,opt,iter,i,j,k,fnzmax,fnzmin,addobj,bigbou,big,l)
c
c Solver finished, return
c
 999  return
      end
```

The interface expects a double precision array (*colnzs*) of size *rsiz*, an integer array (*colidx*) of size *isiz* which represent A and Q. Parameters *mm* and *nn* pass the number of rows and columns in A respectively. Parameter *qnn* contains the number of quadratic variables. For pure LP problems this has to be set to zero, and the allocation of the arrays *qdiag* and *qpnt* can be ignored. Parameter *qnnz* contains the number of off-diagonal nonzeros in the lower triangular part of Q. For separable quadratic problems this has to be set to zero and the allocation of the array *qpnt* can be omitted.

The other input parameters *colpnt, qpnt, ubound, lbound, obj, qdiag, rhs, addobj, big* and the output parameters *xs, dv, dspr, opt, code* are as mentioned previously. An additional output parameter *bstat* is also passed in our sample interface. This parameter is an integer vector of size $(nn + mm)$ which contains 0 or 1 as output corresponding the optimizer's guess on the status of the individual variables / slacks (non-basic – basic respectively).

It is to be noted that no default values for the optimization parameters are defined in our sample interface routine. These parameters will be set from the parameter file by the *readpar* procedure.

To demonstrate the calling the interface routine let us consider our previous sample problem. Then, the input parameters need to be defined as:

```
 mm     = 2
 mn     = 2
 qnn    = 2
 qnnz   = 1
 addobj = 4.0
 isiz   = ...   (size of the allocated integer memory)
 rsiz   = ...   (size of the allocated double prec. memory)

 ubound :  ( 20, big, big,    0)
 lbound :  (  0,   0,   0, -big)
 rhs    :  (  2,   6)
 obj    :  (1.5,  -2)
 qdiag  :  (  8,  10)

colcnt  :  ( 1,       3,       5)
qpnt    :  ( 5,       6,       6)
colidx  :  ( 1, 2,    1, 2,    2)
colnzs  :  ( 2, -1,   1, 2,    2)
```

5 The parameter file

BPMPD is a research code which allowes to experiment by trying different algorithmic options. To support this, several algorithmic parameters and input/output options can be specified for BPMPD via the parameter file called *bpmpd.par*. In the parameter file all parameters must be in lines between the BEGIN and END markers, other lines will be ignored. It is also important that to preserve compatibility across platforms and compilers, EACH numerical value MUST contain a decimal point. Within one line the parameter name and its associated value has to be separated with at least one space or with one "=" symbol. BPMPD will ignore the section of a line which is after a "!" symbol, this allows to put comments in the parameter section.

5.1 Input/output options

The purpose of these options is to support MPS file handling. They can be omitted in the callable library form of BPMPD.

parameter : NAME default: values : String

This option specifies the name of the input file. If it is left blank (as default), BPMPD will prompt the user for the filename. The lenght of the string is limited to 36 characters. The extension "*.mps*" is automatically added to the string read in. If the opening of the file is not successful, BPMPD replaces the extension with"*.qps*" and tries again.

parameter : STARTSOL default: values : String

This string specifies the name of the file which contains the starting point used in warm start. The length of this string is limited to 40 characters. The whole filename (with extension) has to be given. BPMPD can read a file which has been generated by BPMPD itself previousely with OUTPUT = 1 setting (see parameter OUTPUT) or which has the same file format as BPMPD's output file. If the value of the parameter is left blank, no file will be read. If reading the file is unsuccessful, BPMPD will automatically switch from warm start to cold start.

parameter : MAXMN default: 250000 values : POSITIVE INTEGER

BPMPD will partition the memory and read the MPS file by assuming that the number of rows plus number of columns is less than MAXMN. If this is incorrect, BPMPD will repartition the memory and read the MPS file once more. After successful reading the MPS file the dimensions of the problem will be known and BPMPD will redistribute the memory for its maximal possible exploitation. It is recommended to give an overestimation on the problem dimension otherwise BPMPD will need to read the MPS file twice. If the MAXMN value is too large to form the corresponding memory partitions BPMPD will automatically decrease it.

parameter : MINMAX default: 1 values : -1 or 1

BPMPD solves minimization problems. For maximization problems MINMAX has to be specified as -1, then BPMPD will negate the objective function (and minimize it).

parameter : OUTPUT default:1 values : 0 or 1

This parameter defines either a whole solution report has to be written or not. When specifying 1, BPMPD will create an output file which contains the whole optimal solution of the problem. Otherwise a short file will be created containing the most important statistics about the solution progress.

parameter : OBJNAM default: values : String of 8 character length

This parameter specifies the name of the objective function in the MPS file. If it is left blank, BPMPD will select the first row which has the type " N ".

parameter : RHSNAM default: values : String of 8 character length

This parameter specifies the name of the right hand side in the MPS file. If it is left blank, BPMPD will select the first right hand side name.

parameter : BNDNAM default: values : String of 8 character length

This parameter specifies the name of the bounds in the MPS file. If it is left blank, BPMPD will select the first bound name.

parameter : RNGNAM default: values : String of 8 character length

This parameter specifies the name of the ranges in the MPS file. If it is left blank, BPMPD will select the first range name.

parameter : BIGBOUND default:10^{15} values : POSITIVE REAL

All bounds and ranges will be considered as infinity which are larger than this parameter. This parameter serves to correct those MPS files in which infinities are expressed by very large numbers.

parameter : SMALLVAL default:10^{-12} values : POSITIVE REAL

All numerical values in the MPS file which are less in absolute value than this parameter will be considered as zero.

parameter : EXPSLACKS default:0 values : 0 or 1

BPMPD will include slack variables as structural ones (and form the constrains as equalities) if this parameter is set to 1. Otherwise slacks are handled as special variables for inequality constraints.

parameter : ITERLOG default:3 values : NONNEGATIVE INTEGER

This value specifies how the iteration log has to be produced. The following values are possible:

0 : BPMPD works in silent mode, no log will be created.

1 : BPMPD sends the iteration log to the standard output.

2 : BPMPD sends the iteration log to the logfile (which has the same name as the input file but with "*.log*" extension).

3 : BPMPD sends the iteration log to both the standard output and log file.

Larger numbers can be also specified for debugging purposes. In this case BPMPD will print additional informations during the solution process.

5.2 Parameters for handling sparsity

One of the most important features of BPMPD is the sophisticated methods for handling sparsity. Two different approaches are implemented to build the factorization for computing interior point iterations. One is called as *normal equations* while the other is called as *augmented system* approach. The main difference between them is that by the normal equations the pivot order is computed by symbolical investigation while by the augmented system it is determined by both symbolical and numerical tests. The augmented system is more advantageous if the problem has special structures (e.g. dense columns) but it is more expensive than the normal equations approach. See more details in [4]. The following parameters influence the heuristic which decides which of the two approaches is preferable by the particular problems. The implemented method uses an improved version of the heuristic described in [4].

parameter : DENSGAP default: 0.20 values : POSITIVE REAL

Maximal ratio of the dense columns. At most $DENSGAP \cdot n$ columns are allowed to be handled as dense.

parameter : DENSLEN default: 10 values : POSITIVE INTEGER

Minimum required column length for dense columns. Shorter columns are not treated as dense.

parameter : SUPDENS default: 350 values : POSITIVE INTEGER

Minimal column length of 'super' dense columns. Very dense columns are handled specially in the factorization set–up phase to increase the speed of the process.

parameter : LAM **default: 10^{-5}** **values : REAL**

This parameter represents the minimum acceptable density for "dense" columns. This parameter can override the result of the heuristic in such a case if the heuristic decides to handle some columns as "dense" which are of very small density. Positive LAM means a relative density measure while a negative value is negated and considered as absolute density value (i.e. column count).

parameter : SETLAM **default: 0** **values : -1, 0, 1**

By default, BPMPD uses its own heuristic to identify dense columns. When specifying -1, BPMPD skips this heuristic and uses LAM as criterion to handle columns as "dense" and "sparse". Setting up the factorization by the augmented system approach can be requested by specifying 1 for this parameter.
We would like to note that to request the normal equations approach is possible when specifying 1.0 for LAM.

5.3 Supernode parameters

BPMPD performs the supernodal Cholesy factorization. A paper [8] is available which contains the particulars. The parameters introduced here control the supernode partitions of the factorization.

parameter : PSUPNODE **default: 4** **values : POSITIVE INTEGER**

Required minimum number of columns of a primer supernode.

parameter : SSUPNODE **default: 4** **values : POSITIVE INTEGER**

Required minimum number of columns of a secondary supernode.

parameter : MAXSNZ **default: 99999999** **values : POSITIVE INTEGER**

Maximum number of nonzeros allowed in one supernode. This parameter can be very important if the computer platform has an efficient cache memory system.

5.4 Pivot and factorization parameters

These parameters control the pivot searching in the symbolical and numerical processes in the factorization set up as well during interior point iterations.

parameter : TPIV1 **default: 10^{-3}** **values : POSITIVE REAL**

First relative pivot tolerance for the augmented system factorization.

parameter : TPIV2 default: 10^{-8} values : POSITIVE REAL

Second relative pivot tolerance for the augmented system factorization. If there are no pivots satisfying the first tolerance, the value of the tolerance will be reduced until TPIV2. See [7] for more details.

parameter : TABS default: 10^{-12} values : POSITIVE REAL

Relative pivot tolerance for the starting factorization. When a computed pivot value in the factorization is less than this parameter, the corresponding row of the problem will be considered as dependent and will be dropped. This parameter effects the first factorization which is performed to compute the starting point.

parameter : TRABS default: 10^{-15} values : POSITIVE REAL

Relative pivot tolerance during the algorithm. Pivot values bellow this parameter will indicate dependent rows.

parameter : TFIND default: 25 values : NONNEGATIVE INTEGER

Pivot searching power. TFIND = 0 will result in the "minimum degree" ordering while a large number will result in the "minimum local fill–in" ordering. Intermediate values balances between speed and ordering quality. See more details in [6].

parameter : ORDERING default: 2 values : 0, 1, 2 or 3.

This parameter selects the ordering algorithm. The following vales are possible:

0 : "Natural" ordering (without investigations).

1 : Minimum degree ordering

2 : Minimum local fill–in ordering for the normal equations, minimum degree for the augmented system.

3 : Minimum local fill–in ordering.

5.5 Stopping criterion parameters

BPMPD terminates with optimal status if the relative primal infeasibility

$$\frac{||Ax-b||}{1+||b||},$$

the relative dual infeasibility

$$\frac{||A^Ty+z-w-c||}{1+||c+Qx||}$$

and the relative duality gap

$$\frac{|c^Tx+\frac{1}{2}x^TQx-(b^Ty-u^Tw-\frac{1}{2}x^TQx)|}{1+|b^Ty-u^Tw-\frac{1}{2}x^TQx|}$$

or the average complementarity gap

$$\frac{x^Ts}{n}$$

are less than the predetermined tolerances. Infeasibility is detected if a sharp increase in one of the above terms occurs. This is monitored by the *merit function*:

$$\frac{||Ax-b||}{1+||b||}+\frac{||A^Ty+z-w-c||}{1+||c+Qx||}+\frac{|c^Tx+\frac{1}{2}x^TQx-(b^Ty-u^Tw-\frac{1}{2}x^TQx)|}{1+|b^Ty-u^Tw-\frac{1}{2}x^TQx|}.$$

parameter : TOPT1 default: 10^{-8} values : POSITIVE REAL

BPMPD stops if the current iterate is primal and dual feasible and the relative duality gap is less than this value.

parameter : TOPT2 default: 10^{-20} values : POSITIVE REAL

BPMPD stops if the average complementary gap is less than this tolerance.

parameter : TFEAS1 default: 10^{-7} values : POSITIVE REAL

Relative primal feasibility tolerance.

parameter : TFEAS2 default: 10^{-7} values : POSITIVE REAL

Relative dual feasibility tolerance.

parameter : INFTOL **default: 10^5** **values : POSITIVE REAL**
BPMPD stops if the "merit" value of "non-optimality" grows faster than this value.

parameter : TSDIR **default: 10^{-16}** **values : POSITIVE REAL**
BPMPD stops if the maximum norm of the search direction is less than this value.

parameter : MAXITER **default: 99** **values : POSITIVE INTEGER**
BPMPD stops if the number of iterations exceeds this limit.

5.6 Numerical tolerances

These parameters are used in special parts of the code. It is not recommended to change these values.

parameter : TPLUS **default: 10^{-10}** **values : POSITIVE REAL**
Relative addition tolerance. In some procedures (for example in the aggregator) the result of $a + b$ is replaced by zero if $abs\,(a + b) < \max(a, b) \cdot TPLUS$.

parameter : TZER **default: 10^{-35}** **values : POSITIVE REAL**
Absolute zero tolerance. All computed values whose absolute value is less than this parameter will be considered as zero.

5.7 Iterative refinement parameters

During computation of the search direction iterative methods can be invoked to improve numerical accuracy. Three such methods are available in BPMPD, these are:

- Iterative improvement.
- Conjugate gradient method.
- Quasi minimum residual algorithm.

The application of refinements can slow down the iterations remarkably if the problem and its factorization is very sparse. Otherwise, it has less influence on the speed. These methods are highly recommendable due to their advantageous numerical effect.

parameter : TRESX **default: 10^{-9}** **values : POSITIVE REAL**
Acceptable residual in the primal space.

parameter : TRESY **default: 10^{-9}** **values : POSITIVE REAL**
Acceptable residual in the dual space. Iterative refinement will be performed if the primal or dual residual is not acceptable.

parameter : MAXREF **default: 5** **values : INTEGER**
Base number for computing the maximum allowable refinement steps. If its value is positive, the current limit of the iterative refinements is computed as

$$MAXREF \cdot \log_2 \left(\frac{flops}{fnz} \right)$$

where *flops* is the number of floating point operations required for one factorization and *fnz* is the number of nonzeroes in the factorization.
If a negative value is presented, BPMPD will do at most –MAXREF refinements.

parameter : REFMET **default: 1** **values : 0 to 7** **INTEGER**
The bits of this value represent turning on–off individual refinement techniques. The following methods are available:

1 Iterative improvement.

2 Conjugate gradient method.

4 Quasi minimum residual method.

5.8 Scaling parameters

Scaling is performed prior to the interior point iterations to improve the condition and numerical behavior of the problem. The scaling is performed in iterations (called here as *passes*). There are the following scaling methods available:

0 No scaling.

1 Simple scaling. Scales each columns and rows to have 1.0 maximum norm.

2 Geometric mean scaling + simple scaling.

3 Curtis-Reid's algorithm + simple scaling.

4 Geometric mean scaling only.

5 Curtis-Reid's algorithm only.

There are two possible places for scaling: before the aggregator (which is recommended) and after aggregator (which is not recommended).

parameter : OBJNORM default: 10^2 values : NONNEGATIVE REAL
BPMPD will scale the objective to this maximum norm. If the problem is quadratic, the objective will be scaled in such a way that the square of the largest diagonal element of Q will be OBJNORM. If the value of OBJNORM is zero, no scaling on the objective will be performed.

parameter : RHSNORM default: 0.0 values : NONNEGATIVE REAL
BPMPD will scale the right hand side to this maximum norm. If this value is zero, no scaling on the right hand side will be performed.

parameter : MINDIFF default: 1.0 values : POSITIVE REAL
BPMPD will terminate the scaling if the ratio of the largest and smallest absolute value in the matrix is less than this threshold.

parameter : SIGNORE default: 10^{-12} values : POSITIVE REAL
BPMPD will ignore those matrix elements during scaling whose relative absolute value to the maximum norm of the corresponding column is less than SIGNORE. This parameter is very useful if small computational errors are presented in the problem.

parameter : SPASSES1 default: 5 values : NONNEGATIVE INTEGER

Maximum number of passes before the aggregator. This value should be less than 128.

parameter : SMETHOD1 default: 2 values : 0 to 5 INTEGER
Scaling method before aggregator.

parameter : SPASSES2 default: 0 values : NONNEGATIVE INTEGER

Maximum number of passes after aggregator. This value should be less than 128. We do not recommend to scale after doing aggregation, but it can be advantageous in some cases.

parameter : SMETHOD2 default: 0 values : 0 to 5 INTEGER
Scaling method after aggregator.

5.9 Predictor-corrector and barrier parameters

The parameters defined here influence the predictor–corrector method [5] as well the computation of the barrier parameter.

parameter : STOPCOR default: 10^{-3} values : POSITIVE REAL
Corrector is ignored if it decreases the steplengts by a larger factor than STOPCOR.

parameter : BARSET default: 0.25 values : NONNEGATIVE REAL
$BARSET \cdot \frac{x^T s}{n}$ will be used as an upper limit for the barrier parameter.

parameter : BARGROW default: 10^3 values : POSITIVE REAL
Barrier growing bound between two successive iterations.

parameter : BARMIN default: 10^{-12} values : NONNEGATIVE REAL
Minimum barrier threshold. Smaller barrier value will be replaced by zero.

parameter : MINCORR default: 1 values : -1, 0 or 1
Number of minimum corrector steps in the predictor–corrector procedure. Value 1 forces to do one correction with iterative refinement, while values 0 and -1 allow to skip corrector. When value 0 is specified the predictor is enhanced with iterative refinement.

parameter : MAXCORR default: 1 values : NONNEGATIVE INTEGER
Maximum number of corrections allowed in the predictor–corrector procedure.

parameter : INIBARR default: 0.0 values : NONNEGATIVE REAL
This parameter determines the barrier value for the predictor direction as $INIBARR \cdot \frac{x^T s}{n}$.

5.10 Centrality corrections parameters

After the predictor–corrector step BPMPD uses centrality corrections. More details in [3].

parameter : TARGET default: 0.09 values : POSITIVE REAL
Trial steplength improvement.

parameter : TSMALL default: 0.1 values : POSITIVE REAL
Small complementarity bound.

parameter : TLARGE default: 10.0 values : POSITIVE REAL
Large complementarity bound.

parameter : CENTER default: 5.0 values : POSITIVE REAL
Centrality force.

parameter : CORSTOP default: 1.01 values : POSITIVE REAL
Correction stop parameter.

parameter : MINCCORR default: 0 values : NONNEGATIVE INTEGER
Number of the minimum corrections.

parameter : MAXCCORR default: 9 values : NONNEGATIVE INTEGER
Base number for computing the number of maximum corrections. The number of maximum corrections will be computed similarly as described by the parameter MAXREF.

5.11 Steplenth parameters

Both by LP an QP problems different stepsizes are used in the primal and dual spaces. In the quadratic case, however, a special technique is implemented which guarantees the decrease in the primal and dual infeasibilities. The following parameters are used to prevent to hit the boundary during interior point iterations. Smaller values can be beneficial on "difficult" problems.

parameter : PRDARE default: 0.999 values : 0.0 < PRDARE < 1.0

The maximal possible steplengt to the boundary in the primal space is multiplied by PRDARE.

parameter : DUDARE default: 0.999 values : 0.0 < DUDARE < 1.0

The maximal possible steplengt to the boundary in the dual space is multiplied by DUDARE.

5.12 Starting point paramerers

There are two possibilities for determination of the starting point: cold start and warm start (under developing). By cold start BPMPD computes for the primal variables the solution of the problem:

$$\min \ ||\hat{x}|| + P \cdot \hat{x}^T Q \hat{x},$$

$$A\hat{x} = b$$

where $P > 0$ weight for the quadratic objective function. For the dual variables there are two options. Either the optimal solution of the problem

$$\min \ ||\hat{s}|| + P \cdot \hat{s}^T Q \hat{s}$$

$$A^T y + \hat{s} - Q\hat{x} = c$$

or $\hat{y} = 0$, $\hat{s} = c + Q\hat{x}$ is computed. To avoid small or negative values, $\hat{x}$ and $\hat{s}$ are modified (*safed*).

parameter : PRMIN default: 150.0 values : POSITIVE REAL

Minimum initial primal value. This parameter is used during the modification of $\hat{x}$ and $\hat{s}$.

parameter : UPMAX default: 50000.0 values : POSITIVE REAL

Maximum initial primal value. This parameter is used during the modification of $\hat{x}$ and $\hat{s}$.

parameter : DUMIN default: 150.0 values : POSITIVE REAL

Minimum initial dual slack value. This parameter is used during the modification of $\hat{x}$ and $\hat{s}$.

parameter : LSQWEIGHT default: 10.0 values : POSITIVE REAL
Weight of $||\hat{x}||$ by the primal problem.

parameter : SMETHOD default: 2 values : -2, -1, 1 or 2
By this parameter the first or second method for the dual can be selected. When negative value is presented, BPMPD try warm start. If the warm start is not successful, BPMPD will use -SMETHOD in the cold start process.

parameter : SAFEMET default: -3 values : from -3 to +3
This parameter selects the method for modification of $\hat{x}$ and $\hat{s}$. Negative values implies methods which improves all components otherwise only the small components are modified. Method 1 (and -1) checks the individual values of the variables while methods 2 and 3 (as well -2 and -3) consider the complementarity components also.

parameter : REGULARIZE default: 0 values : 0 or 1
Value 1. introduces artificial variables (slack variables for equality rows with upper and lower bounds zero) to the model.

5.13 Presolve parameters

Prior to the solution process BPMPD may apply presolve techniques. These techniques reduce the problem size by detecting fix variables or redundant rows, and in general, improve the efficiency of the interior point method. Several presolve techniques can be invoked. In addition to the presolve, BPMPD is able to eliminate variables and constraints from the LP/QP problem. A paper is available which discuss the particulars [9].

parameter : PRESOLV default: 1023 values : 0 to 2047 INTEGER
Presolve action control. The bits of this value represent turning on or off individual presolve techniques. The following methods are available:

1 : Singleton row check. Singleton rows are replaced by bounds on variables.

2 : Singleton column check. Free (or implied free) singleton columns are eliminated from the problem.

4 : Primal feasibility check. This process evaluates the possible minimum and maximum row values. Based on the results, it may detect redundant rows and fixes variables on their bound.

8 : Cheap dual test. This procedure performs tests based on the signs of the nonzero values of the columns.

16 : Dual feasibility check. The same as the primal but on the dual problem. Bounds on the dual variables are tightened if possible.

32 : Primal bound check and relaxation. This procedure detects hidden free variables in the problem.

64 : Searching identical variables. This procedure detects splitted variables in the problem and replaces them with an appropriate one.

128 : Doubleton row check. This procedure generates free variables in doubleton rows.

256 : Aggregator. This procedure eliminates free variables.

512 : Sparser. This procedure makes the constraint matrix sparser.

1024 : Extended dual test. This procedure performs additional dual tests which may change the optimal solution. The optimal objective value is not change however, therefore this procedure can be turned on if the optimal objective function value is the only important.

parameter : BNDLOOP default: 5 values : NONNEGATIVE INTEGER

Maximum number of iterations in the primal bound check procedure.

parameter : DULOOP default: 10 values : NONNEGATIVE INTEGER

Maximum number of iterations in the dual feasibility check procedure.

parameter : PRIMALBND default: 1000.0 values : NONNEGATIVE REAL

Maximum allowable generated bound during primal bound investigations.

parameter : DUALBND default: 10000.0 values : NONNEGATIVE REAL

Maximum allowable generated bound during dual feasibility tests.

parameter : PREFEAS default: 10^{-8} values : NONNEGATIVE REAL
Relative primal feasibility tolerance during presolve.

parameter : PIVRTOL default: 10^{-2} values : NONNEGATIVE REAL
Relative pivot tolerance in the aggregator process.

parameter : PIVATOL default: 10^{-4} values : NONNEGATIVE REAL
Absolute pivot tolerance in the aggregator process.

parameter : FILLTOL default: 4.0 values : NONNEGATIVE REAL
Fill–in bound on individual pivots in aggregator. Aggregator stops if

$$(c_i - 1)(r_j - 1) > FILLTOL \cdot (c_i + r_j)$$

for all possible pivot positions, where (i, j) are the column/row indices of the pivot candidates and c_i, r_j are the column and row counter of column i and row j, respectively.

5.14 Regularization parameters

The purpose of these parameters is to prevent numerical instability. Unfortunately, increasing numerical behavior (e.g. doing stronger regularization) may decrease the efficiency of the computed steps and possibly increases the number of necessary iterations. In extreme cases regularization can prevent convergence. The theory of regularization is discussed in [9].

parameter : SOFTREG default: 10^{-12} values : NONNEGATIVE REAL
Soft regularization of the diagonal scaling matrix. This parameter slowes down the decrease of small values in the diagonal scaling matrix. The increase of this parameter will increase the regularization.

parameter : HARDREG default: 10^{-16} values : NONNEGATIVE REAL
The parameter introduces "hard" regularization, namely "freeses" small values in the diagonal scaling matrix. The increase of this parameter will increase the regularization.

parameter : SCFREE default: 10^{-8} values : NONNEGATIVE REAL

Base value for scaling free variables. See [9] for the details.

parameter : TFIXVAR default: 10^{-20} values : NONNEGATIVE REAL
BPMPD fixes variables whose value fall under this limit.

parameter : TFIXSLACK default: 10^{-20} values : NONNEGATIVE REAL
BPMPD fixes slacks whose value fall under this limit.

parameter : DSLACKLIM default: 10^{-20} values : NONNEGATIVE REAL
Dual slack limit. BPMPD does not decrease dual slacks bellow this value.

parameter : COMPLIMIT default: 1.0 values : NONNEGATIVE REAL
Complementarity balancing technique turns on if the average complementarity falls bellow this value.

parameter : COMPPAR default: 10^{-5} values : NONNEGATIVE REAL
The complementarity balancing technique will improve those complementarity components which are less then the average complementarity gap multiplied by COMPPAR.

6 Bug reports and further inquires

If there are any problems or questions about the BPMPD package please contact:

Csaba Mészáros,
Computer and Automation Research Institute,
Laboratory of Operations Research and Decision Systems,
H-1518 Budapest,
P.O. BOX 63, Hungary .
FAX: +(361) 269-8268
PHONE: +(361) 269-8267
E-MAIL: mcsaba@oplab.sztaki.hu, meszaros@sztaki.hu

References

[1] T.J. Carpenter, I.J. Lustig, J.M. Mulvey, and D.F. Shanno. Higher order predictor-corrector interior point methods with application to quadratic objectives. *SIAM Journal on Optim.*, 3:696–725, 1993.

[2] A.V. Fiacco and G.P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* John Wiley and Sons, New York, 1968.

[3] J. Gondzio. Multiple centrality corrections in a primal-dual method for linear programming. *Computational Optimization and Applications*, 6:137–156, 1996.

[4] I. Maros and Cs. Mészáros. The role of the augmented system in interior point methods. Technical Report TR/06/95, Brunel University, Department of Mathematics and Statistics, London, 1995. to appear in European Journal of Operations Research.

[5] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optim.*, 2(4):575–601, 1992.

[6] Cs. Mészáros. The "inexact" minimum local fill–in ordering algorithm. Working paper WP 95–7, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, 1995.

[7] Cs. Mészáros. The augmented system variant of IPMs in two–stage stochastic linear programming computation. Working paper WP 95–1, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, 1995. to appear in European Journal of Operations Research.

[8] Cs. Mészáros. Fast Cholesky factorization for interior point methods of linear programming. *Computers & Mathematics with Applications*, 31(4/5):49–54, 1996.

[9] Cs. Mészáros. On free variables in interior point methods. Technical Report DOC 97/4, Imperial College, Department of Computing, London, 1997.

[10] R. D. C. Monteiro and I. Adler. Interior path following primal-dual algorithms. Part II: Convex quadratic programming. *Math. Programming*, 44:43–66, 1989.

[11] J.L. Nazareth. *Computer Solution of Linear Programs.* Oxford University Press, New York, 1987.