

14.3 Parameters grouped by topic

Basis identification

- *simLuTolRelPiv*
- *biCleanOptimizer*
- *biIgnoreMaxIter*
- *biIgnoreNumError*
- *biMaxIterations*
- *intpntBasis*
- *logBi*
- *logBiFreq*

Conic interior-point method

- *intpntCoTolDfeas*
- *intpntCoTolInfeas*
- *intpntCoTolMuRed*
- *intpntCoTolNearRel*
- *intpntCoTolPfeas*
- *intpntCoTolRelGap*

Data input/output

- *logFile*
- *writeLpFullObj*
- *writeLpLineWidth*
- *writeLpQuotedNames*
- *writeLpTermsPerLine*
- *basSolFileName*
- *dataFileName*
- *intSolFileName*
- *itrSolFileName*
- *writeLpGenVarName*

Dual simplex

- *simDualCrash*
- *simDualRestrictSelection*
- *simDualSelection*

Infeasibility report

- *logInfeasAna*

Interior-point method

- *intpntCoTolDfeas*
- *intpntCoTolInfeas*
- *intpntCoTolMuRed*
- *intpntCoTolNearRel*
- *intpntCoTolPfeas*
- *intpntCoTolRelGap*
- *intpntTolDfeas*
- *intpntTolDsafe*
- *intpntTolInfeas*
- *intpntTolMuRed*
- *intpntTolPath*
- *intpntTolPfeas*
- *intpntTolPsafe*
- *intpntTolRelGap*
- *intpntTolRelStep*
- *intpntTolStepSize*
- *biIgnoreMaxIter*
- *biIgnoreNumError*
- *intpntBasis*
- *intpntDiffStep*
- *intpntMaxIterations*
- *intpntMaxNumCor*
- *intpntOffColTrh*
- *intpntOrderGpNumSeeds*
- *intpntOrderMethod*
- *intpntRegularizationUse*
- *intpntScaling*
- *intpntSolveForm*
- *intpntStartingPoint*
- *logIntpnt*

- *mioBranchDir*
- *mioConicOuterApproximation*
- *mioCutClique*
- *mioCutCmir*
- *mioCutGmi*
- *mioCutImpliedBound*
- *mioCutKnapsackCover*
- *mioCutSelectionLevel*
- *mioFeaspumpLevel*
- *mioHeuristicLevel*
- *mioMaxNumBranches*
- *mioMaxNumRelaxs*
- *mioMaxNumRootCutRounds*
- *mioMaxNumSolutions*
- *mioNodeOptimizer*
- *mioNodeSelection*
- *mioPerspectiveReformulate*
- *mioProbingLevel*
- *mioPropagateObjectiveConstraint*
- *mioRinsMaxNodes*
- *mioRootOptimizer*
- *mioRootRepeatPresolveLevel*
- *mioSeed*
- *mioVbDetectionLevel*

Output information

- *licenseSuppressExpireWrns*
- *licenseTrhExpiryWrn*
- *log*
- *logBi*
- *logBiFreq*
- *logCutSecondOpt*
- *logExpand*
- *logFile*
- *logInfeasAna*
- *logIntpnt*

Solution input/output

- *basSolFileName*
- *intSolFileName*
- *itrSolFileName*

Termination criteria

- *basisRelTolS*
- *basisTolS*
- *basisTolX*
- *intpntCoTolDfeas*
- *intpntCoTolInfeas*
- *intpntCoTolMuRed*
- *intpntCoTolNearRel*
- *intpntCoTolPfeas*
- *intpntCoTolRelGap*
- *intpntTolDfeas*
- *intpntTolInfeas*
- *intpntTolMuRed*
- *intpntTolPfeas*
- *intpntTolRelGap*
- *lowerObjCut*
- *lowerObjCutFiniteTrh*
- *mioMaxTime*
- *mioRelGapConst*
- *mioTolRelGap*
- *optimizerMaxTime*
- *upperObjCut*
- *upperObjCutFiniteTrh*
- *biMaxIterations*
- *intpntMaxIterations*
- *mioMaxNumBranches*
- *mioMaxNumRootCutRounds*
- *mioMaxNumSolutions*
- *simMaxIterations*

15.1 The LP File Format

MOSEK supports the LP file format with some extensions. The LP format is not a completely well-defined standard and hence different optimization packages may interpret the same LP file in slightly different ways. **MOSEK** tries to emulate as closely as possible CPLEX's behavior, but tries to stay backward compatible.

The LP file format can specify problems of the form

$$\begin{array}{ll} \text{minimize/maximize} & c^T x + \frac{1}{2} q^o(x) \\ \text{subject to} & \begin{array}{lll} l^c \leq & Ax + \frac{1}{2} q(x) & \leq u^c, \\ l^x \leq & x & \leq u^x, \\ & x_{\mathcal{J}} \text{ integer,} \end{array} \end{array}$$

where

- $x \in \mathbb{R}^n$ is the vector of decision variables.
- $c \in \mathbb{R}^n$ is the linear term in the objective.
- $q^o : \mathbb{R}^n \rightarrow \mathbb{R}$ is the quadratic term in the objective where

$$q^o(x) = x^T Q^o x$$

and it is assumed that

$$Q^o = (Q^o)^T.$$

- $A \in \mathbb{R}^{m \times n}$ is the constraint matrix.
- $l^c \in \mathbb{R}^m$ is the lower limit on the activity for the constraints.
- $u^c \in \mathbb{R}^m$ is the upper limit on the activity for the constraints.
- $l^x \in \mathbb{R}^n$ is the lower limit on the activity for the variables.
- $u^x \in \mathbb{R}^n$ is the upper limit on the activity for the variables.
- $q : \mathbb{R}^n \rightarrow \mathbb{R}$ is a vector of quadratic functions. Hence,

$$q_i(x) = x^T Q^i x$$

where it is assumed that

$$Q^i = (Q^i)^T.$$

- $\mathcal{J} \subseteq \{1, 2, \dots, n\}$ is an index set of the integer constrained variables.

15.1.1 File Sections

An LP formatted file contains a number of sections specifying the objective, constraints, variable bounds, and variable types. The section keywords may be any mix of upper and lower case letters.

Objective Function

The first section beginning with one of the keywords

```
max
maximum
maximize
min
minimum
minimize
```

defines the objective sense and the objective function, i.e.

$$c^T x + \frac{1}{2} x^T Q^o x.$$

The objective may be given a name by writing

```
myname:
```

before the expressions. If no name is given, then the objective is named **obj**.

The objective function contains linear and quadratic terms. The linear terms are written as

```
4 x1 + x2 - 0.1 x3
```

and so forth. The quadratic terms are written in square brackets (`[]/2`) and are either squared or multiplied as in the examples

```
x1^2
```

and

```
x1 * x2
```

There may be zero or more pairs of brackets containing quadratic expressions.

An example of an objective section is

```
minimize
myobj: 4 x1 + x2 - 0.1 x3 + [ x1^2 + 2.1 x1 * x2 ]/2
```

Please note that the quadratic expressions are multiplied with $\frac{1}{2}$, so that the above expression means

$$\text{minimize } 4x_1 + x_2 - 0.1 \cdot x_3 + \frac{1}{2}(x_1^2 + 2.1 \cdot x_1 \cdot x_2)$$

If the same variable occurs more than once in the linear part, the coefficients are added, so that `4 x1 + 2 x1` is equivalent to `6 x1`. In the quadratic expressions `x1 * x2` is equivalent to `x2 * x1` and, as in the linear part, if the same variables multiplied or squared occur several times their coefficients are added.

Constraints

The second section beginning with one of the keywords

```
subj to
subject to
s.t.
st
```

defines the linear constraint matrix A and the quadratic matrices Q^i .

A constraint contains a name (optional), expressions adhering to the same rules as in the objective and a bound:

```
subject to
con1: x1 + x2 + [ x3^2 ]/2 <= 5.1
```

The bound type (here `<=`) may be any of `<`, `<=`, `=`, `>`, `>=` (`<` and `<=` mean the same), and the bound may be any number.

In the standard LP format it is not possible to define more than one bound per line, but **MOSEK** supports defining ranged constraints by using double-colon (`::`) instead of a single-colon (`:`) after the constraint name, i.e.

$$-5 \leq x_1 + x_2 \leq 5 \tag{15.1}$$

may be written as

```
con:: -5 < x_1 + x_2 < 5
```

By default **MOSEK** writes ranged constraints this way.

If the files must adhere to the LP standard, ranged constraints must either be split into upper bounded and lower bounded constraints or be written as an equality with a slack variable. For example the expression (15.1) may be written as

$$x_1 + x_2 - sl_1 = 0, \quad -5 \leq sl_1 \leq 5.$$

Bounds

Bounds on the variables can be specified in the bound section beginning with one of the keywords

```
bound
bounds
```

The bounds section is optional but should, if present, follow the **subject to** section. All variables listed in the bounds section must occur in either the objective or a constraint.

The default lower and upper bounds are 0 and $+\infty$. A variable may be declared free with the keyword **free**, which means that the lower bound is $-\infty$ and the upper bound is $+\infty$. Furthermore it may be assigned a finite lower and upper bound. The bound definitions for a given variable may be written in one or two lines, and bounds can be any number or $\pm\infty$ (written as **+inf/-inf/+infinity/-infinity**) as in the example

```
bounds
x1 free
x2 <= 5
0.1 <= x2
x3 = 42
2 <= x4 < +inf
```

Variable Types

The final two sections are optional and must begin with one of the keywords

```
bin
binaries
binary
```

and

```
gen
general
```

Under **general** all integer variables are listed, and under **binary** all binary (integer variables with bounds 0 and 1) are listed:

```
general
x1 x2
binary
x3 x4
```

Again, all variables listed in the binary or general sections must occur in either the objective or a constraint.

Terminating Section

Finally, an LP formatted file must be terminated with the keyword

```
end
```

15.1.2 LP File Examples

Linear example `lo1.lp`

```
\ File: lo1.lp
maximize
obj: 3 x1 + x2 + 5 x3 + x4
subject to
c1: 3 x1 + x2 + 2 x3 = 30
c2: 2 x1 + x2 + 3 x3 + x4 >= 15
c3: 2 x2 + 3 x4 <= 25
bounds
0 <= x1 <= +infinity
0 <= x2 <= 10
0 <= x3 <= +infinity
0 <= x4 <= +infinity
end
```

Mixed integer example `mil01.lp`

```
maximize
obj: x1 + 6.4e-01 x2
subject to
c1: 5e+01 x1 + 3.1e+01 x2 <= 2.5e+02
c2: 3e+00 x1 - 2e+00 x2 >= -4e+00
bounds
0 <= x1 <= +infinity
0 <= x2 <= +infinity
general
x1 x2
end
```

15.1.3 LP Format peculiarities

Comments

Anything on a line after a `\` is ignored and is treated as a comment.

Names

A name for an objective, a constraint or a variable may contain the letters `a-z`, `A-Z`, the digits `0-9` and the characters

```
!"#$%&()/,.;?@_`'|~
```

The first character in a name must not be a number, a period or the letter `e` or `E`. Keywords must not be used as names.

MOSEK accepts any character as valid for names, except `\0`. A name that is not allowed in LP file will be changed and a warning will be issued.

The algorithm for making names LP valid works as follows: The name is interpreted as an `utf-8` string. For a Unicode character `c`:

- If `c==_` (underscore), the output is `__` (two underscores).
- If `c` is a valid LP name character, the output is just `c`.
- If `c` is another character in the ASCII range, the output is `_XX`, where `XX` is the hexadecimal code for the character.
- If `c` is a character in the range `127-65535`, the output is `_uXXXX`, where `XXXX` is the hexadecimal code for the character.

- If c is a character above 65535, the output is `_XXXXXXXX`, where `XXXXXXXX` is the hexadecimal code for the character.

Invalid `utf-8` substrings are escaped as `_XX'`, and if a name starts with a period, `e` or `E`, that character is escaped as `_XX`.

Variable Bounds

Specifying several upper or lower bounds on one variable is possible but **MOSEK** uses only the tightest bounds. If a variable is fixed (with `=`), then it is considered the tightest bound.

MOSEK Extensions to the LP Format

Some optimization software packages employ a more strict definition of the LP format than the one used by **MOSEK**. The limitations imposed by the strict LP format are the following:

- Quadratic terms in the constraints are not allowed.
- Names can be only 16 characters long.
- Lines must not exceed 255 characters in length.

To get around some of the inconveniences converting from other problem formats, **MOSEK** allows lines to contain 1024 characters and names may have any length (shorter than the 1024 characters).

Formatting of an LP File

A few parameters control the visual formatting of LP files written by **MOSEK** in order to make it easier to read the files. These parameters are

- `writeLpLineWidth` sets the maximum number of characters on a single line. The default value is 80 corresponding roughly to the width of a standard text document.
- `writeLpTermsPerLine` sets the maximum number of terms per line; a term means a sign, a coefficient, and a name (for example `+ 42 elephants`). The default value is 0, meaning that there is no maximum.

Unnamed Constraints

Reading and writing an LP file with **MOSEK** may change it superficially. If an LP file contains unnamed constraints or objective these are given their generic names when the file is read (however unnamed constraints in **MOSEK** are written without names).

15.2 The MPS File Format

MOSEK supports the standard MPS format with some extensions. For a detailed description of the MPS format see the book by Nazareth [Naz87].

15.2.1 MPS File Structure

The version of the MPS format supported by **MOSEK** allows specification of an optimization problem of the form

$$\begin{aligned}
 &\text{maximize/minimize} && c^T x + q_0(x) \\
 &l^c \leq && Ax + q(x) \leq u^c, \\
 &l^x \leq && x \leq u^x, \\
 &&& x \in \mathcal{K}, \\
 &&& x_{\mathcal{I}} \text{ integer},
 \end{aligned} \tag{15.2}$$

where

- $x \in \mathbb{R}^n$ is the vector of decision variables.

- $A \in \mathbb{R}^{m \times n}$ is the constraint matrix.
- $l^c \in \mathbb{R}^m$ is the lower limit on the activity for the constraints.
- $u^c \in \mathbb{R}^m$ is the upper limit on the activity for the constraints.
- $l^x \in \mathbb{R}^n$ is the lower limit on the activity for the variables.
- $u^x \in \mathbb{R}^n$ is the upper limit on the activity for the variables.
- $q : \mathbb{R}^n \rightarrow \mathbb{R}$ is a vector of quadratic functions. Hence,

$$q_i(x) = \frac{1}{2}x^T Q^i x$$

where it is assumed that $Q^i = (Q^i)^T$. Please note the explicit $\frac{1}{2}$ in the quadratic term and that Q^i is required to be symmetric. The same applies to q_0 .

- \mathcal{K} is a convex cone.
- $\mathcal{J} \subseteq \{1, 2, \dots, n\}$ is an index set of the integer-constrained variables.
- c is the vector of objective coefficients.

An MPS file with one row and one column can be illustrated like this:

```
*          1          2          3          4          5          6
*23456789012345678901234567890123456789012345678901234567890
NAME          [name]
OBJSENSE
    [objsense]
OBJNAME        [objname]
ROWS
    ?  [cname1]
COLUMNS
    [vname1]  [cname1]  [value1]          [cname2]  [value2]
RHS
    [name]    [cname1]  [value1]          [cname2]  [value2]
RANGES
    [name]    [cname1]  [value1]          [cname2]  [value2]
QSECTION      [cname1]
    [vname1]  [vname2]  [value1]          [vname3]  [value2]
QMATRIX
    [vname1]  [vname2]  [value1]
QUADOBJ
    [vname1]  [vname2]  [value1]
QCMATRIX      [cname1]
    [vname1]  [vname2]  [value1]
BOUNDS
    ?? [name]  [vname1]  [value1]
CSECTION      [kname1]  [value1]          [ktype]
    [vname1]
ENDATA
```

Here the names in capitals are keywords of the MPS format and names in brackets are custom defined names or values. A couple of notes on the structure:

- Fields: All items surrounded by brackets appear in *fields*. The fields named “valueN” are numerical values. Hence, they must have the format

$[+|-] \text{XXXXXXX} . \text{XXXXXX} [[e|E] [+|-] \text{XXX}]$

where

$x = [0|1|2|3|4|5|6|7|8|9] .$

- Sections: The MPS file consists of several sections where the names in capitals indicate the beginning of a new section. For example, **COLUMNS** denotes the beginning of the columns section.
- Comments: Lines starting with an ***** are comment lines and are ignored by **MOSEK**.
- Keys: The question marks represent keys to be specified later.
- Extensions: The sections **QSECTION** and **CSECTION** are specific **MOSEK** extensions of the MPS format. The sections **QMATRIX**, **QUADOBJ** and **QCMATRIX** are included for sake of compatibility with other vendors extensions to the MPS format.
- The standard MPS format is a fixed format, i.e. everything in the MPS file must be within certain fixed positions. **MOSEK** also supports a *free format*. See [Sec. 15.2.5](#) for details.

Linear example lo1.mps

A concrete example of a MPS file is presented below:

```
* File: lo1.mps
NAME          lo1
OBJSENSE
    MAX
ROWS
N  obj
E  c1
G  c2
L  c3
COLUMNS
    x1      obj      3
    x1      c1       3
    x1      c2       2
    x2      obj      1
    x2      c1       1
    x2      c2       1
    x2      c3       2
    x3      obj      5
    x3      c1       2
    x3      c2       3
    x4      obj      1
    x4      c2       1
    x4      c3       3
RHS
    rhs     c1      30
    rhs     c2      15
    rhs     c3      25
RANGES
BOUNDS
UP bound    x2      10
ENDATA
```

Subsequently each individual section in the MPS format is discussed.

NAME (optional)

In this section a name (**[name]**) is assigned to the problem.

OBJSENSE (optional)

This is an optional section that can be used to specify the sense of the objective function. The **OBJSENSE** section contains one line at most which can be one of the following:

MIN
MINIMIZE
MAX
MAXIMIZE

It should be obvious what the implication is of each of these four lines.

OBJNAME (optional)

This is an optional section that can be used to specify the name of the row that is used as objective function. `objname` should be a valid row name.

ROWS

A record in the ROWS section has the form

? [cname1]

where the requirements for the fields are as follows:

Field	Starting Position	Max Width	required	Description
?	2	1	Yes	Constraint key
[cname1]	5	8	Yes	Constraint name

Hence, in this section each constraint is assigned a unique name denoted by [cname1]. Please note that [cname1] starts in position 5 and the field can be at most 8 characters wide. An initial key ? must be present to specify the type of the constraint. The key can have values E, G, L, or N with the following interpretation:

Constraint type	l_i^c	u_i^c
E (equal)	finite	$= l_i^c$
G (greater)	finite	∞
L (lower)	$-\infty$	finite
N (none)	$-\infty$	∞

In the MPS format the objective vector is not specified explicitly, but one of the constraints having the key N will be used as the objective vector c . In general, if multiple N type constraints are specified, then the first will be used as the objective vector c , unless something else was specified in the section OBJNAME.

COLUMNS

In this section the elements of A are specified using one or more records having the form:

[vname1] [cname1] [value1] [cname2] [value2]

where the requirements for each field are as follows:

Field	Starting Position	Max Width	required	Description
[vname1]	5	8	Yes	Variable name
[cname1]	15	8	Yes	Constraint name
[value1]	25	12	Yes	Numerical value
[cname2]	40	8	No	Constraint name
[value2]	50	12	No	Numerical value

Hence, a record specifies one or two elements a_{ij} of A using the principle that [vname1] and [cname1] determines j and i respectively. Please note that [cname1] must be a constraint name specified in the ROWS section. Finally, [value1] denotes the numerical value of a_{ij} . Another optional element is specified by [cname2], and [value2] for the variable specified by [vname1]. Some important comments are:

- All elements belonging to one variable must be grouped together.
- Zero elements of A should not be specified.

- At least one element for each variable should be specified.

RHS (optional)

A record in this section has the format

[name]	[cname1]	[value1]	[cname2]	[value2]
--------	----------	----------	----------	----------

where the requirements for each field are as follows:

Field	Starting Position	Max Width	required	Description
[name]	5	8	Yes	Name of the RHS vector
[cname1]	15	8	Yes	Constraint name
[value1]	25	12	Yes	Numerical value
[cname2]	40	8	No	Constraint name
[value2]	50	12	No	Numerical value

The interpretation of a record is that [name] is the name of the RHS vector to be specified. In general, several vectors can be specified. [cname1] denotes a constraint name previously specified in the ROWS section. Now, assume that this name has been assigned to the i -th constraint and v_1 denotes the value specified by [value1], then the interpretation of v_1 is:

Constraint	l_i^c	u_i^c
E	v_1	v_1
G	v_1	
L		v_1
N		

An optional second element is specified by [cname2] and [value2] and is interpreted in the same way. Please note that it is not necessary to specify zero elements, because elements are assumed to be zero.

RANGES (optional)

A record in this section has the form

[name]	[cname1]	[value1]	[cname2]	[value2]
--------	----------	----------	----------	----------

where the requirements for each fields are as follows:

Field	Starting Position	Max Width	required	Description
[name]	5	8	Yes	Name of the RANGE vector
[cname1]	15	8	Yes	Constraint name
[value1]	25	12	Yes	Numerical value
[cname2]	40	8	No	Constraint name
[value2]	50	12	No	Numerical value

The records in this section are used to modify the bound vectors for the constraints, i.e. the values in l^c and u^c . A record has the following interpretation: [name] is the name of the RANGE vector and [cname1] is a valid constraint name. Assume that [cname1] is assigned to the i -th constraint and let v_1 be the value specified by [value1], then a record has the interpretation:

Constraint type	Sign of v_1	l_i^c	u_i^c
E	—	$u_i^c + v_1$	
E	+		$l_i^c + v_1$
G	— or +		$l_i^c + v_1 $
L	— or +	$u_i^c - v_1 $	
N			

Another constraint bound can optionally be modified using [cname2] and [value2] the same way.

QSECTION (optional)

Within the QSECTION the label [cname1] must be a constraint name previously specified in the ROWS section. The label [cname1] denotes the constraint to which the quadratic terms belong. A record in the QSECTION has the form

[vname1]	[vname2]	[value1]	[vname3]	[value2]
----------	----------	----------	----------	----------

where the requirements for each field are:

Field	Starting Position	Max Width	required	Description
[vname1]	5	8	Yes	Variable name
[vname2]	15	8	Yes	Variable name
[value1]	25	12	Yes	Numerical value
[vname3]	40	8	No	Variable name
[value2]	50	12	No	Numerical value

A record specifies one or two elements in the lower triangular part of the Q^i matrix where [cname1] specifies the i . Hence, if the names [vname1] and [vname2] have been assigned to the k -th and j -th variable, then Q_{kj}^i is assigned the value given by [value1]. An optional second element is specified in the same way by the fields [vname1], [vname3], and [value2].

The example

$$\begin{array}{ll} \text{minimize} & -x_2 + \frac{1}{2}(2x_1^2 - 2x_1x_3 + 0.2x_2^2 + 2x_3^2) \\ \text{subject to} & x_1 + x_2 + x_3 \geq 1, \\ & x \geq 0 \end{array}$$

has the following MPS file representation

```
* File: qo1.mps
NAME          qo1
ROWS
  N  obj
  G  c1
COLUMNS
  x1      c1      1.0
  x2      obj     -1.0
  x2      c1      1.0
  x3      c1      1.0
RHS
  rhs     c1      1.0
QSECTION    obj
  x1      x1      2.0
  x1      x3     -1.0
  x2      x2      0.2
  x3      x3      2.0
ENDATA
```

Regarding the QSECTIONS please note that:

- Only one QSECTION is allowed for each constraint.
- The QSECTIONS can appear in an arbitrary order after the COLUMNS section.
- All variable names occurring in the QSECTION must already be specified in the COLUMNS section.
- All entries specified in a QSECTION are assumed to belong to the lower triangular part of the quadratic term of Q .

QMATRIX/QUADOBJ (optional)

The QMATRIX and QUADOBJ sections allow to define the quadratic term of the objective function. They differ in how the quadratic term of the objective function is stored:

- **QMATRIX** stores all the nonzeros coefficients, without taking advantage of the symmetry of the Q matrix.
- **QUADOBJ** stores the upper diagonal nonzero elements of the Q matrix.

A record in both sections has the form:

[vname1]	[vname2]	[value1]
----------	----------	----------

where the requirements for each field are:

Field	Starting Position	Max Width	required	Description
[vname1]	5	8	Yes	Variable name
[vname2]	15	8	Yes	Variable name
[value1]	25	12	Yes	Numerical value

A record specifies one elements of the Q matrix in the objective function . Hence, if the names [vname1] and [vname2] have been assigned to the k -th and j -th variable, then Q_{kj} is assigned the value given by [value1]. Note that a line must appear for each off-diagonal coefficient if using a **QMATRIX** section, while only one entry is required in a **QUADOBJ** section. The quadratic part of the objective function will be evaluated as $1/2x^T Qx$.

The example

$$\begin{aligned}
 &\text{minimize} && -x_2 + \frac{1}{2}(2x_1^2 - 2x_1x_3 + 0.2x_2^2 + 2x_3^2) \\
 &\text{subject to} && x_1 + x_2 + x_3 \geq 1, \\
 &&& x \geq 0
 \end{aligned}$$

has the following MPS file representation using **QMATRIX**

```

* File: qo1_matrix.mps
NAME                qo1_qmatrix
ROWS
  N  obj
  G  c1
COLUMNS
  x1      c1      1.0
  x2      obj     -1.0
  x2      c1      1.0
  x3      c1      1.0
RHS
  rhs     c1      1.0
QMATRIX
  x1      x1      2.0
  x1      x3     -1.0
  x3      x1     -1.0
  x2      x2      0.2
  x3      x3      2.0
ENDATA

```

or the following using **QUADOBJ**

```

* File: qo1_quadobj.mps
NAME                qo1_quadobj
ROWS
  N  obj
  G  c1
COLUMNS
  x1      c1      1.0
  x2      obj     -1.0
  x2      c1      1.0
  x3      c1      1.0
RHS
  rhs     c1      1.0

```

(continues on next page)

```

QUADOBJ
  x1      x1      2.0
  x1      x3     -1.0
  x2      x2      0.2
  x3      x3      2.0
ENDATA

```

Please also note that:

- A QMATRIX/QUADOBJ section can appear in an arbitrary order after the COLUMNS section.
- All variable names occurring in the QMATRIX/QUADOBJ section must already be specified in the COLUMNS section.

QMATRIX (optional)

A QMATRIX section allows to specify the quadratic part of a given constraint. Within the QMATRIX the label [cname1] must be a constraint name previously specified in the ROWS section. The label [cname1] denotes the constraint to which the quadratic term belongs. A record in the QSECTION has the form

```
[vname1] [vname2] [value1]
```

where the requirements for each field are:

Field	Starting Position	Max Width	required	Description
[vname1]	5	8	Yes	Variable name
[vname2]	15	8	Yes	Variable name
[value1]	25	12	Yes	Numerical value

A record specifies an entry of the Q^i matrix where [cname1] specifies the i . Hence, if the names [vname1] and [vname2] have been assigned to the k -th and j -th variable, then Q_{kj}^i is assigned the value given by [value1]. Moreover, the quadratic term is represented as $1/2x^T Qx$.

The example

$$\begin{array}{ll}
 \text{minimize} & x_2 \\
 \text{subject to} & x_1 + x_2 + x_3 \geq 1, \\
 & \frac{1}{2}(-2x_1x_3 + 0.2x_2^2 + 2x_3^2) \leq 10, \\
 & x \geq 0
 \end{array}$$

has the following MPS file representation

```

* File: qo1.mps
NAME          qo1
ROWS
  N  obj
  G  c1
  L  q1
COLUMNS
  x1      c1      1.0
  x2      obj     -1.0
  x2      c1      1.0
  x3      c1      1.0
RHS
  rhs     c1      1.0
  rhs     q1     10.0
QMATRIX
  q1
  x1      x1      2.0
  x1      x3     -1.0
  x3      x1     -1.0
  x2      x2      0.2
  x3      x3      2.0
ENDATA

```

Regarding the QCMATRIXs please note that:

- Only one QCMATRIX is allowed for each constraint.
- The QCMATRIXs can appear in an arbitrary order after the COLUMNS section.
- All variable names occurring in the QSECTION must already be specified in the COLUMNS section.
- QCMATRIX does not exploit the symmetry of Q : an off-diagonal entry (i, j) should appear twice.

BOUNDS (optional)

In the BOUNDS section changes to the default bounds vectors l^x and u^x are specified. The default bounds vectors are $l^x = 0$ and $u^x = \infty$. Moreover, it is possible to specify several sets of bound vectors. A record in this section has the form

??	[name]	[vname1]	[value1]
----	--------	----------	----------

where the requirements for each field are:

Field	Starting Position	Max Width	Required	Description
??	2	2	Yes	Bound key
[name]	5	8	Yes	Name of the BOUNDS vector
[vname1]	15	8	Yes	Variable name
[value1]	25	12	No	Numerical value

Hence, a record in the BOUNDS section has the following interpretation: [name] is the name of the bound vector and [vname1] is the name of the variable for which the bounds are modified by the record. ?? and [value1] are used to modify the bound vectors according to the following table:

??	l_j^x	u_j^x	Made integer (added to \mathcal{J})
FR	$-\infty$	∞	No
FX	v_1	v_1	No
LO	v_1	unchanged	No
MI	$-\infty$	unchanged	No
PL	unchanged	∞	No
UP	unchanged	v_1	No
BV	0	1	Yes
LI	$\lceil v_1 \rceil$	unchanged	Yes
UI	unchanged	$\lfloor v_1 \rfloor$	Yes

Here v_1 is the value specified by [value1].

CSECTION (optional)

The purpose of the CSECTION is to specify the conic constraint

$$x \in \mathcal{K}$$

in (15.2). It is assumed that \mathcal{K} satisfies the following requirements. Let

$$x^t \in \mathbb{R}^{n^t}, \quad t = 1, \dots, k$$

be vectors comprised of parts of the decision variables x so that each decision variable is a member of exactly **one** vector x^t , for example

$$x^1 = \begin{bmatrix} x_1 \\ x_4 \\ x_7 \end{bmatrix} \quad \text{and} \quad x^2 = \begin{bmatrix} x_6 \\ x_5 \\ x_3 \\ x_2 \end{bmatrix}.$$

Next define

$$\mathcal{K} := \{x \in \mathbb{R}^n : \quad x^t \in \mathcal{K}_t, \quad t = 1, \dots, k\}$$

where \mathcal{K}_t must have one of the following forms:

- \mathbb{R} set:

$$\mathcal{K}_t = \mathbb{R}^{n^t}.$$

- Zero cone:

$$\mathcal{K}_t = \{0\} \subseteq \mathbb{R}^{n^t}. \quad (15.3)$$

- Quadratic cone:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\}. \quad (15.4)$$

- Rotated quadratic cone:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : 2x_1x_2 \geq \sum_{j=3}^{n^t} x_j^2, \quad x_1, x_2 \geq 0 \right\}. \quad (15.5)$$

- Primal exponential cone:

$$\mathcal{K}_t = \{x \in \mathbb{R}^3 : x_1 \geq x_2 \exp(x_3/x_2), \quad x_1, x_2 \geq 0\}. \quad (15.6)$$

- Primal power cone (with parameter $0 < \alpha < 1$):

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : x_1^\alpha x_2^{1-\alpha} \geq \sqrt{\sum_{j=3}^{n^t} x_j^2}, \quad x_1, x_2 \geq 0 \right\}. \quad (15.7)$$

- Dual exponential cone:

$$\mathcal{K}_t = \{x \in \mathbb{R}^3 : x_1 \geq -x_3 e^{-1} \exp(x_2/x_3), \quad x_3 \leq 0, x_1 \geq 0\}. \quad (15.8)$$

- Dual power cone (with parameter $0 < \alpha < 1$):

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : \left(\frac{x_1}{\alpha}\right)^\alpha \left(\frac{x_2}{1-\alpha}\right)^{1-\alpha} \geq \sqrt{\sum_{j=3}^{n^t} x_j^2}, \quad x_1, x_2 \geq 0 \right\}. \quad (15.9)$$

In general, membership in the \mathbb{R} set is not specified. If a variable is not a member of any other cone then it is assumed to be a member of the \mathbb{R} cone.

Next, let us study an example. Assume that the power cone

$$x_4^{1/3} x_5^{2/3} \geq |x_8|$$

and the rotated quadratic cone

$$2x_3x_7 \geq x_1^2 + x_0^2, \quad x_3, x_7 \geq 0,$$

should be specified in the MPS file. One **CSECTION** is required for each cone and they are specified as follows:

*	1	2	3	4	5	6
*23456789012345678901234567890123456789012345678901234567890						
CSECTION	konea	3e-1		PPOW		
x4						
x5						
x8						
CSECTION	koneb	0.0		RQUAD		
x7						
x3						
x1						
x0						

In general, a CSECTION header has the format

CSECTION	[kname1]	[value1]	[ktype]
----------	----------	----------	---------

where the requirements for each field are as follows:

Field	Starting Position	Max Width	Required	Description
[kname1]	15	8	Yes	Name of the cone
[value1]	25	12	No	Cone parameter
[ktype]	40		Yes	Type of the cone.

The possible cone type keys are:

[ktype]	Members	[value1]	Interpretation.
ZERO	≥ 0	unused	Zero cone (15.3).
QUAD	≥ 1	unused	Quadratic cone (15.4).
RQUAD	≥ 2	unused	Rotated quadratic cone (15.5).
PEXP	3	unused	Primal exponential cone (15.6).
PPOW	≥ 2	α	Primal power cone (15.7).
DEXP	3	unused	Dual exponential cone (15.8).
DPOW	≥ 2	α	Dual power cone (15.9).

A record in the CSECTION has the format

[vname1]

where the requirements for each field are

Field	Starting Position	Max Width	required	Description
[vname1]	5	8	Yes	A valid variable name

A variable must occur in at most one CSECTION.

ENDATA

This keyword denotes the end of the MPS file.

15.2.2 Integer Variables

Using special bound keys in the BOUNDS section it is possible to specify that some or all of the variables should be integer-constrained i.e. be members of \mathcal{J} . However, an alternative method is available. This method is available only for backward compatibility and we recommend that it is not used. This method requires that markers are placed in the COLUMNS section as in the example:

COLUMNS				
x1	obj	-10.0	c1	0.7
x1	c2	0.5	c3	1.0
x1	c4	0.1		
* Start of integer-constrained variables.				
MARK000	'MARKER'		'INTORG'	
x2	obj	-9.0	c1	1.0
x2	c2	0.8333333333	c3	0.66666667
x2	c4	0.25		
x3	obj	1.0	c6	2.0
MARK001	'MARKER'		'INTEND'	
* End of integer-constrained variables.				

Please note that special marker lines are used to indicate the start and the end of the integer variables. Furthermore be aware of the following

- All variables between the markers are assigned a default lower bound of 0 and a default upper bound of 1. **This may not be what is intended.** If it is not intended, the correct bounds should be defined in the BOUNDS section of the MPS formatted file.

- **MOSEK** ignores field 1, i.e. **MARK0001** and **MARK001**, however, other optimization systems require them.
- Field 2, i.e. **MARKER**, must be specified including the single quotes. This implies that no row can be assigned the name **MARKER**.
- Field 3 is ignored and should be left blank.
- Field 4, i.e. **INTORG** and **INTEND**, must be specified.
- It is possible to specify several such integer marker sections within the **COLUMNS** section.

15.2.3 General Limitations

- An MPS file should be an ASCII file.

15.2.4 Interpretation of the MPS Format

Several issues related to the MPS format are not well-defined by the industry standard. However, **MOSEK** uses the following interpretation:

- If a matrix element in the **COLUMNS** section is specified multiple times, then the multiple entries are added together.
- If a matrix element in a **QSECTION** section is specified multiple times, then the multiple entries are added together.

15.2.5 The Free MPS Format

MOSEK supports a free format variation of the MPS format. The free format is similar to the MPS file format but less restrictive, e.g. it allows longer names. However, a name must not contain any blanks.

15.3 The OPF Format

The *Optimization Problem Format (OPF)* is an alternative to LP and MPS files for specifying optimization problems. It is row-oriented, inspired by the CPLEX LP format.

Apart from containing objective, constraints, bounds etc. it may contain complete or partial solutions, comments and extra information relevant for solving the problem. It is designed to be easily read and modified by hand and to be forward compatible with possible future extensions.

Intended use

The OPF file format is meant to replace several other files:

- The LP file format: Any problem that can be written as an LP file can be written as an OPF file too; furthermore it naturally accommodates ranged constraints and variables as well as arbitrary characters in names, fixed expressions in the objective, empty constraints, and conic constraints.
- Parameter files: It is possible to specify integer, double and string parameters along with the problem (or in a separate OPF file).
- Solution files: It is possible to store a full or a partial solution in an OPF file and later reload it.

15.3.1 The File Format

The format uses tags to structure data. A simple example with the basic sections may look like this:

```

[comment]
This is a comment. You may write almost anything here...
[/comment]

# This is a single-line comment.

[objective min 'myobj']
x + 3 y + x^2 + 3 y^2 + z + 1
[/objective]

[constraints]
[con 'con01'] 4 <= x + y  [/con]
[/constraints]

[bounds]
[b] -10 <= x,y <= 10  [/b]

[cone quad] x,y,z [/cone]
[/bounds]

```

A scope is opened by a tag of the form `[tag]` and closed by a tag of the form `[/tag]`. An opening tag may accept a list of unnamed and named arguments, for examples:

```

[tag value] tag with one unnamed argument [/tag]
[tag arg=value] tag with one named argument [/tag]

```

Unnamed arguments are identified by their order, while named arguments may appear in any order, but never before an unnamed argument. The `value` can be a quoted, single-quoted or double-quoted text string, i.e.

```

[tag 'value']      single-quoted value [/tag]
[tag arg='value']  single-quoted value [/tag]
[tag "value"]      double-quoted value [/tag]
[tag arg="value"]  double-quoted value [/tag]

```

15.3.2 Sections

The recognized tags are

`[comment]`

A comment section. This can contain *almost* any text: Between single quotes (') or double quotes (") any text may appear. Outside quotes the markup characters ([and]) must be prefixed by backslashes. Both single and double quotes may appear alone or inside a pair of quotes if it is prefixed by a backslash.

`[objective]`

The objective function: This accepts one or two parameters, where the first one (in the above example `min`) is either `min` or `max` (regardless of case) and defines the objective sense, and the second one (above `myobj`), if present, is the objective name. The section may contain linear and quadratic expressions.

If several objectives are specified, all but the last are ignored.

`[constraints]`

This does not directly contain any data, but may contain subsections `con` defining a linear constraint.

`[con]`

Defines a single constraint; if an argument is present (`[con NAME]`) this is used as the name of the constraint, otherwise it is given a null-name. The section contains a constraint definition written as

linear and quadratic expressions with a lower bound, an upper bound, with both or with an equality. Examples:

```
[constraints]
[con 'con1'] 0 <= x + y      [/con]
[con 'con2'] 0 >= x + y      [/con]
[con 'con3'] 0 <= x + y <= 10 [/con]
[con 'con4']      x + y = 10 [/con]
[/constraints]
```

Constraint names are unique. If a constraint is specified which has the same name as a previously defined constraint, the new constraint replaces the existing one.

[bounds]

This does not directly contain any data, but may contain subsections **b** (linear bounds on variables) and **cone** (cones).

[b]

Bound definition on one or several variables separated by comma (,). An upper or lower bound on a variable replaces any earlier defined bound on that variable. If only one bound (upper or lower) is given only this bound is replaced. This means that upper and lower bounds can be specified separately. So the OPF bound definition:

```
[b] x,y >= -10 [/b]
[b] x,y <= 10  [/b]
```

results in the bound $-10 \leq x, y \leq 10$.

[cone]

Specifies a cone. A cone is defined as a sequence of variables which belong to a single unique cone. The supported cone types are:

- **quad**: a quadratic cone of n variables x_1, \dots, x_n defines a constraint of the form

$$x_1^2 \geq \sum_{i=2}^n x_i^2, \quad x_1 \geq 0.$$

- **rquad**: a rotated quadratic cone of n variables x_1, \dots, x_n defines a constraint of the form

$$2x_1x_2 \geq \sum_{i=3}^n x_i^2, \quad x_1, x_2 \geq 0.$$

- **pexp**: primal exponential cone of 3 variables x_1, x_2, x_3 defines a constraint of the form

$$x_1 \geq x_2 \exp(x_3/x_2), \quad x_1, x_2 \geq 0.$$

- **ppow** with parameter $0 < \alpha < 1$: primal power cone of n variables x_1, \dots, x_n defines a constraint of the form

$$x_1^\alpha x_2^{1-\alpha} \geq \sqrt{\sum_{j=3}^n x_j^2}, \quad x_1, x_2 \geq 0.$$

- **dexp**: dual exponential cone of 3 variables x_1, x_2, x_3 defines a constraint of the form

$$x_1 \geq -x_3 e^{-1} \exp(x_2/x_3), \quad x_3 \leq 0, x_1 \geq 0.$$

- **dpow** with parameter $0 < \alpha < 1$: dual power cone of n variables x_1, \dots, x_n defines a constraint of the form

$$\left(\frac{x_1}{\alpha}\right)^\alpha \left(\frac{x_2}{1-\alpha}\right)^{1-\alpha} \geq \sqrt{\sum_{j=3}^n x_j^2}, \quad x_1, x_2 \geq 0.$$

- **zero**: zero cone of n variables x_1, \dots, x_n defines a constraint of the form

$$x_1 = \dots = x_n = 0$$

A `[bounds]`-section example:

```
[bounds]
[b] 0 <= x,y <= 10 [/b] # ranged bound
[b] 10 >= x,y >= 0 [/b] # ranged bound
[b] 0 <= x,y <= inf [/b] # using inf
[b]      x,y free [/b] # free variables
# Let (x,y,z,w) belong to the cone K
[cone rquad] x,y,z,w [/cone] # rotated quadratic cone
[cone ppow '3e-01' 'a'] x1, x2, x3 [/cone] # power cone with alpha=1/3 and name 'a'
[/bounds]
```

By default all variables are free.

`[variables]`

This defines an ordering of variables as they should appear in the problem. This is simply a space-separated list of variable names.

`[integer]`

This contains a space-separated list of variables and defines the constraint that the listed variables must be integer-valued.

`[hints]`

This may contain only non-essential data; for example estimates of the number of variables, constraints and non-zeros. Placed before all other sections containing data this may reduce the time spent reading the file.

In the `hints` section, any subsection which is not recognized by **MOSEK** is simply ignored. In this section a hint is defined as follows:

```
[hint ITEM] value [/hint]
```

The hints recognized by **MOSEK** are:

- **numvar** (number of variables),
- **numcon** (number of linear/quadratic constraints),
- **numanz** (number of linear non-zeros in constraints),
- **numqnz** (number of quadratic non-zeros in constraints).

`[solutions]`

This section can contain a set of full or partial solutions to a problem. Each solution must be specified using a `[solution]`-section, i.e.

```
[solutions]
[solution]...[/solution] #solution 1
[solution]...[/solution] #solution 2
#other solutions....
[solution]...[/solution] #solution n
[/solutions]
```

The syntax of a [solution]-section is the following:

```
[solution SOLTYPE status=STATUS]...[/solution]
```

where SOLTYPE is one of the strings

- interior, a non-basic solution,
- basic, a basic solution,
- integer, an integer solution,

and STATUS is one of the strings

- UNKNOWN,
- OPTIMAL,
- INTEGER_OPTIMAL,
- PRIM_FEAS,
- DUAL_FEAS,
- PRIM_AND_DUAL_FEAS,
- NEAR_OPTIMAL,
- NEAR_PRIM_FEAS,
- NEAR_DUAL_FEAS,
- NEAR_PRIM_AND_DUAL_FEAS,
- PRIM_INFEAS_CER,
- DUAL_INFEAS_CER,
- NEAR_PRIM_INFEAS_CER,
- NEAR_DUAL_INFEAS_CER,
- NEAR_INTEGER_OPTIMAL.

Most of these values are irrelevant for input solutions; when constructing a solution for simplex hot-start or an initial solution for a mixed integer problem the safe setting is UNKNOWN.

A [solution]-section contains [con] and [var] sections. Each [con] and [var] section defines solution information for a single variable or constraint, specified as list of KEYWORD/value pairs, in any order, written as

```
KEYWORD=value
```

Allowed keywords are as follows:

- sk. The status of the item, where the value is one of the following strings:
 - LOW, the item is on its lower bound.
 - UPR, the item is on its upper bound.
 - FIX, it is a fixed item.
 - BAS, the item is in the basis.

- SUPBAS, the item is super basic.
 - UNK, the status is unknown.
 - INF, the item is outside its bounds (infeasible).
- **lvl** Defines the level of the item.
 - **s1** Defines the level of the dual variable associated with its lower bound.
 - **su** Defines the level of the dual variable associated with its upper bound.
 - **sn** Defines the level of the variable associated with its cone.
 - **y** Defines the level of the corresponding dual variable (for constraints only).

A **[var]** section should always contain the items **sk**, **lvl**, **s1** and **su**. Items **s1** and **su** are not required for **integer** solutions.

A **[con]** section should always contain **sk**, **lvl**, **s1**, **su** and **y**.

An example of a solution section

```
[solution basic status=UNKNOWN]
[var x0] sk=LOW    lvl=5.0      [/var]
[var x1] sk=UPR    lvl=10.0     [/var]
[var x2] sk=SUPBAS lvl=2.0    s1=1.5 su=0.0 [/var]

[con c0] sk=LOW    lvl=3.0 y=0.0 [/con]
[con c0] sk=UPR    lvl=0.0 y=5.0 [/con]
[/solution]
```

- **[vendor]** This contains solver/vendor specific data. It accepts one argument, which is a vendor ID – for **MOSEK** the ID is simply **mosek** – and the section contains the subsection **parameters** defining solver parameters. When reading a vendor section, any unknown vendor can be safely ignored. This is described later.

Comments using the **#** may appear anywhere in the file. Between the **#** and the following line-break any text may be written, including markup characters.

15.3.3 Numbers

Numbers, when used for parameter values or coefficients, are written in the usual way by the **printf** function. That is, they may be prefixed by a sign (+ or -) and may contain an integer part, decimal part and an exponent. The decimal point is always **.** (a dot). Some examples are

```
1
1.0
.0
1.
1e10
1e+10
1e-10
```

Some *invalid* examples are

```
e10    # invalid, must contain either integer or decimal part
.       # invalid
.e10   # invalid
```

More formally, the following standard regular expression describes numbers as used:

```
[+|-]?([0-9]+[.][0-9]*)|([eE][+|-]?[0-9]+)?
```

15.3.4 Names

Variable names, constraint names and objective name may contain arbitrary characters, which in some cases must be enclosed by quotes (single or double) that in turn must be preceded by a backslash. Unquoted names must begin with a letter (a-z or A-Z) and contain only the following characters: the letters a-z and A-Z, the digits 0-9, braces ({ and }) and underscore (_).

Some examples of legal names:

```
an_unquoted_name
another_name{123}
'single quoted name'
"double quoted name"
"name with \"quote\" in it"
"name with []s in it"
```

15.3.5 Parameters Section

In the **vendor** section solver parameters are defined inside the **parameters** subsection. Each parameter is written as

```
[p PARAMETER_NAME] value [/p]
```

where **PARAMETER_NAME** is replaced by a **MOSEK** parameter name, usually of the form **MSK_IPAR_...**, **MSK_DPAR_...** or **MSK_SPAR_...**, and the **value** is replaced by the value of that parameter; both integer values and named values may be used. Some simple examples are

```
[vendor mosek]
[parameters]
[p MSK_IPAR_OPF_MAX_TERMS_PER_LINE] 10      [/p]
[p MSK_IPAR_OPF_WRITE_PARAMETERS]    MSK_ON  [/p]
[p MSK_DPAR_DATA_TOL_BOUND_INF]     1.0e18  [/p]
[/parameters]
[/vendor]
```

15.3.6 Writing OPF Files from MOSEK

To write an OPF file add the **.opf** extension to the file name.

15.3.7 Examples

This section contains a set of small examples written in OPF and describing how to formulate linear, quadratic and conic problems.

Linear Example lo1.opf

Consider the example:

$$\begin{array}{llllll} \text{maximize} & 3x_0 & + & 1x_1 & + & 5x_2 & + & 1x_3 \\ \text{subject to} & 3x_0 & + & 1x_1 & + & 2x_2 & & = & 30, \\ & 2x_0 & + & 1x_1 & + & 3x_2 & + & 1x_3 & \geq & 15, \\ & & & 2x_1 & & & + & 3x_3 & \leq & 25, \end{array}$$

having the bounds

$$\begin{array}{llll} 0 & \leq & x_0 & \leq & \infty, \\ 0 & \leq & x_1 & \leq & 10, \\ 0 & \leq & x_2 & \leq & \infty, \\ 0 & \leq & x_3 & \leq & \infty. \end{array}$$

In the OPF format the example is displayed as shown in [Listing 15.1](#).

Listing 15.1: Example of an OPF file for a linear problem.

```
[comment]
  The lo1 example in OPF format
[/comment]

[hints]
  [hint NUMVAR] 4 [/hint]
  [hint NUMCON] 3 [/hint]
  [hint NUMANZ] 9 [/hint]
[/hints]

[variables disallow_new_variables]
  x1 x2 x3 x4
[/variables]

[objective maximize 'obj']
  3 x1 + x2 + 5 x3 + x4
[/objective]

[constraints]
  [con 'c1'] 3 x1 +   x2 + 2 x3           = 30 [/con]
  [con 'c2'] 2 x1 +   x2 + 3 x3 +   x4 >= 15 [/con]
  [con 'c3']      2 x2           + 3 x4 <= 25 [/con]
[/constraints]

[bounds]
  [b] 0 <= * [/b]
  [b] 0 <= x2 <= 10 [/b]
[/bounds]
```

Quadratic Example qo1.opf

An example of a quadratic optimization problem is

$$\begin{aligned} & \text{minimize} && x_1^2 + 0.1x_2^2 + x_3^2 - x_1x_3 - x_2 \\ & \text{subject to} && 1 \leq x_1 + x_2 + x_3, \\ & && x \geq 0. \end{aligned}$$

This can be formulated in `opf` as shown below.

Listing 15.2: Example of an OPF file for a quadratic problem.

```
[comment]
  The qo1 example in OPF format
[/comment]

[hints]
  [hint NUMVAR] 3 [/hint]
  [hint NUMCON] 1 [/hint]
  [hint NUMANZ] 3 [/hint]
  [hint NUMQNZ] 4 [/hint]
[/hints]

[variables disallow_new_variables]
  x1 x2 x3
[/variables]

[objective minimize 'obj']
  # The quadratic terms are often written with a factor of 1/2 as here,
  # but this is not required.
```

(continues on next page)

(continued from previous page)

```
- x2 + 0.5 ( 2.0 x1 ^ 2 - 2.0 x3 * x1 + 0.2 x2 ^ 2 + 2.0 x3 ^ 2 )
[/objective]

[constraints]
[con 'c1'] 1.0 <= x1 + x2 + x3 [/con]
[/constraints]

[bounds]
[b] 0 <= * [/b]
[/bounds]
```

Conic Quadratic Example cqo1.opf

Consider the example:

$$\begin{aligned} \text{minimize} \quad & x_3 + x_4 + x_5 \\ \text{subject to} \quad & x_0 + x_1 + 2x_2 = 1, \\ & x_0, x_1, x_2 \geq 0, \\ & x_3 \geq \sqrt{x_0^2 + x_1^2}, \\ & 2x_4x_5 \geq x_2^2. \end{aligned}$$

Please note that the type of the cones is defined by the parameter to `[cone ...]`; the content of the `cone`-section is the names of variables that belong to the cone. The resulting OPF file is in [Listing 15.3](#).

Listing 15.3: Example of an OPF file for a conic quadratic problem.

```
[comment]
The cqo1 example in OPF format.
[/comment]

[hints]
[hint NUMVAR] 6 [/hint]
[hint NUMCON] 1 [/hint]
[hint NUMANZ] 3 [/hint]
[/hints]

[variables disallow_new_variables]
x1 x2 x3 x4 x5 x6
[/variables]

[objective minimize 'obj']
x4 + x5 + x6
[/objective]

[constraints]
[con 'c1'] x1 + x2 + 2e+00 x3 = 1e+00 [/con]
[/constraints]

[bounds]
# We let all variables default to the positive orthant
[b] 0 <= * [/b]

# ...and change those that differ from the default
[b] x4,x5,x6 free [/b]

# Define quadratic cone: x4 >= sqrt( x1^2 + x2^2 )
[cone quad 'k1'] x4, x1, x2 [/cone]

# Define rotated quadratic cone: 2 x5 x6 >= x3^2
```

(continues on next page)

```
[cone rquad 'k2'] x5, x6, x3 [/cone]
[/bounds]
```

Mixed Integer Example `mil01.opf`

Consider the mixed integer problem:

$$\begin{aligned} & \text{maximize} && x_0 + 0.64x_1 \\ & \text{subject to} && 50x_0 + 31x_1 \leq 250, \\ & && 3x_0 - 2x_1 \geq -4, \\ & && x_0, x_1 \geq 0 \quad \text{and integer} \end{aligned}$$

This can be implemented in OPF with the file in [Listing 15.4](#).

Listing 15.4: Example of an OPF file for a mixed-integer linear problem.

```
[comment]
  The mil01 example in OPF format
[/comment]

[hints]
  [hint NUMVAR] 2 [/hint]
  [hint NUMCON] 2 [/hint]
  [hint NUMANZ] 4 [/hint]
[/hints]

[variables disallow_new_variables]
  x1 x2
[/variables]

[objective maximize 'obj']
  x1 + 6.4e-1 x2
[/objective]

[constraints]
  [con 'c1'] 5e+1 x1 + 3.1e+1 x2 <= 2.5e+2 [/con]
  [con 'c2'] -4 <= 3 x1 - 2 x2 [/con]
[/constraints]

[bounds]
  [b] 0 <= * [/b]
[/bounds]

[integer]
  x1 x2
[/integer]
```

15.4 The CBF Format

This document constitutes the technical reference manual of the *Conic Benchmark Format* with file extension: `.cbf` or `.CBF`. It unifies linear, second-order cone (also known as conic quadratic) and semidefinite optimization with mixed-integer variables. The format has been designed with benchmark libraries in mind, and therefore focuses on compact and easily parsable representations. The problem structure is separated from the problem data, and the format moreover facilitates benchmarking of hotstart capability through sequences of changes.

15.4.1 How Instances Are Specified

This section defines the spectrum of conic optimization problems that can be formulated in terms of the keywords of the CBF format.

In the CBF format, conic optimization problems are considered in the following form:

$$\begin{aligned} & \min / \max \quad g^{obj} \\ \text{s.t.} \quad & g_i \in \mathcal{K}_i, \quad i \in \mathcal{I}, \\ & G_i \in \mathcal{K}_i, \quad i \in \mathcal{I}^{PSD}, \\ & x_j \in \mathcal{K}_j, \quad j \in \mathcal{J}, \\ & \overline{X}_j \in \mathcal{K}_j, \quad j \in \mathcal{J}^{PSD}. \end{aligned} \tag{15.10}$$

- **Variables** are either scalar variables, x_j for $j \in \mathcal{J}$, or variables, \overline{X}_j for $j \in \mathcal{J}^{PSD}$. Scalar variables can also be declared as integer.
- **Constraints** are affine expressions of the variables, either scalar-valued g_i for $i \in \mathcal{I}$, or matrix-valued G_i for $i \in \mathcal{I}^{PSD}$

$$\begin{aligned} g_i &= \sum_{j \in \mathcal{J}^{PSD}} \langle F_{ij}, X_j \rangle + \sum_{j \in \mathcal{J}} a_{ij} x_j + b_i, \\ G_i &= \sum_{j \in \mathcal{J}} x_j H_{ij} + D_i. \end{aligned}$$

- The **objective function** is a scalar-valued affine expression of the variables, either to be minimized or maximized. We refer to this expression as g^{obj}

$$g^{obj} = \sum_{j \in \mathcal{J}^{PSD}} \langle F_j^{obj}, X_j \rangle + \sum_{j \in \mathcal{J}} a_j^{obj} x_j + b^{obj}.$$

CBF format can represent the following cones \mathcal{K} :

- **Free domain** - A cone in the linear family defined by

$$\{x \in \mathbb{R}^n\}, \text{ for } n \geq 1.$$

- **Positive orthant** - A cone in the linear family defined by

$$\{x \in \mathbb{R}^n \mid x_j \geq 0 \text{ for } j = 1, \dots, n\}, \text{ for } n \geq 1.$$

- **Negative orthant** - A cone in the linear family defined by

$$\{x \in \mathbb{R}^n \mid x_j \leq 0 \text{ for } j = 1, \dots, n\}, \text{ for } n \geq 1.$$

- **Fixpoint zero** - A cone in the linear family defined by

$$\{x \in \mathbb{R}^n \mid x_j = 0 \text{ for } j = 1, \dots, n\}, \text{ for } n \geq 1.$$

- **Quadratic cone** - A cone in the second-order cone family defined by

$$\left\{ \begin{pmatrix} p \\ x \end{pmatrix} \in \mathbb{R} \times \mathbb{R}^{n-1}, p^2 \geq x^T x, p \geq 0 \right\}, \text{ for } n \geq 2.$$

- **Rotated quadratic cone** - A cone in the second-order cone family defined by

$$\left\{ \begin{pmatrix} p \\ q \\ x \end{pmatrix} \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^{n-2}, 2pq \geq x^T x, p \geq 0, q \geq 0 \right\}, \text{ for } n \geq 3.$$

15.4.2 The Structure of CBF Files

This section defines how information is written in the CBF format, without being specific about the type of information being communicated.

All information items belong to exactly one of the three groups of information. These information groups, and the order they must appear in, are:

1. File format.
2. Problem structure.
3. Problem data.

The first group, file format, provides information on how to interpret the file. The second group, problem structure, provides the information needed to deduce the type and size of the problem instance. Finally, the third group, problem data, specifies the coefficients and constants of the problem instance.

Information items

The format is composed as a list of information items. The first line of an information item is the **KEYWORD**, revealing the type of information provided. The second line - of some keywords only - is the **HEADER**, typically revealing the size of information that follows. The remaining lines are the **BODY** holding the actual information to be specified.

KEYWORD BODY
KEYWORD HEADER BODY

The **KEYWORD** determines how each line in the **HEADER** and **BODY** is structured. Moreover, the number of lines in the **BODY** follows either from the **KEYWORD**, the **HEADER**, or from another information item required to precede it.

Embedded hotstart-sequences

A sequence of problem instances, based on the same problem structure, is within a single file. This is facilitated via the **CHANGE** within the problem data information group, as a separator between the information items of each instance. The information items following a **CHANGE** keyword are appending to, or changing (e.g., setting coefficients back to their default value of zero), the problem data of the preceding instance.

The sequence is intended for benchmarking of hotstart capability, where the solvers can reuse their internal state and solution (subject to the achieved accuracy) as warmpoint for the succeeding instance. Whenever this feature is unsupported or undesired, the keyword **CHANGE** should be interpreted as the end of file.

File encoding and line width restrictions

The format is based on the US-ASCII printable character set with two extensions as listed below. Note, by definition, that none of these extensions can be misinterpreted as printable US-ASCII characters:

- A line feed marks the end of a line, carriage returns are ignored.
- Comment-lines may contain unicode characters in UTF-8 encoding.

The line width is restricted to 512 bytes, with 3 bytes reserved for the potential carriage return, line feed and null-terminator.

Integers and floating point numbers must follow the ISO C decimal string representation in the standard C locale. The format does not impose restrictions on the magnitude of, or number of significant digits in numeric data, but the use of 64-bit integers and 64-bit IEEE 754 floating point numbers should be sufficient to avoid loss of precision.

Comment-line and whitespace rules

The format allows single-line comments respecting the following rule:

- Lines having first byte equal to '#' (US-ASCII 35) are comments, and should be ignored. Comments are only allowed between information items.

Given that a line is not a comment-line, whitespace characters should be handled according to the following rules:

- Leading and trailing whitespace characters should be ignored.
 - The separator between multiple pieces of information on one line, is either one or more whitespace characters.
- Lines containing only whitespace characters are empty, and should be ignored. Empty lines are only allowed between information items.

15.4.3 Problem Specification

The problem structure

The problem structure defines the objective sense, whether it is minimization and maximization. It also defines the index sets, \mathcal{J} , \mathcal{J}^{PSD} , \mathcal{I} and \mathcal{I}^{PSD} , which are all numbered from zero, $\{0, 1, \dots\}$, and empty until explicitly constructed.

- **Scalar variables** are constructed in vectors restricted to a conic domain, such as $(x_0, x_1) \in \mathbb{R}_+^2$, $(x_2, x_3, x_4) \in \mathcal{Q}^3$, etc. In terms of the Cartesian product, this generalizes to

$$x \in \mathcal{K}_1^{n_1} \times \mathcal{K}_2^{n_2} \times \dots \times \mathcal{K}_k^{n_k}$$

which in the CBF format becomes:

```
VAR
n k
K1 n1
K2 n2
...
Kk nk
```

where $\sum_i n_i = n$ is the total number of scalar variables. The list of supported cones is found in [Table 15.3](#). Integrality of scalar variables can be specified afterwards.

- **PSD variables** are constructed one-by-one. That is, $X_j \succeq \mathbf{0}^{n_j \times n_j}$ for $j \in \mathcal{J}^{PSD}$, constructs a matrix-valued variable of size $n_j \times n_j$ restricted to be symmetric positive semidefinite. In the CBF format, this list of constructions becomes:

```
PSDVAR
N
n1
n2
...
nN
```

where N is the total number of PSD variables.

- **Scalar constraints** are constructed in vectors restricted to a conic domain, such as $(g_0, g_1) \in \mathbb{R}_+^2$, $(g_2, g_3, g_4) \in \mathcal{Q}^3$, etc. In terms of the Cartesian product, this generalizes to

$$g \in \mathcal{K}_1^{m_1} \times \mathcal{K}_2^{m_2} \times \dots \times \mathcal{K}_k^{m_k}$$

which in the CBF format becomes:

```

CON
m k
K1 m1
K2 m2
..
Kk mk

```

where $\sum_i m_i = m$ is the total number of scalar constraints. The list of supported cones is found in [Table 15.3](#).

- **PSD constraints** are constructed one-by-one. That is, $G_i \succeq \mathbf{0}^{m_i \times m_i}$ for $i \in \mathcal{I}^{PSD}$, constructs a matrix-valued affine expressions of size $m_i \times m_i$ restricted to be symmetric positive semidefinite. In the CBF format, this list of constructions becomes

```

PSDCON
M
m1
m2
..
mM

```

where M is the total number of PSD constraints.

With the objective sense, variables (with integer indications) and constraints, the definitions of the many affine expressions follow in problem data.

Problem data

The problem data defines the coefficients and constants of the affine expressions of the problem instance. These are considered zero until explicitly defined, implying that instances with no keywords from this information group are, in fact, valid. Duplicating or conflicting information is a failure to comply with the standard. Consequently, two coefficients written to the same position in a matrix (or to transposed positions in a symmetric matrix) is an error.

The affine expressions of the objective, g^{obj} , of the scalar constraints, g_i , and of the PSD constraints, G_i , are defined separately. The following notation uses the standard trace inner product for matrices, $\langle X, Y \rangle = \sum_{i,j} X_{ij} Y_{ij}$.

- The affine expression of the objective is defined as

$$g^{obj} = \sum_{j \in \mathcal{J}^{PSD}} \langle F_j^{obj}, X_j \rangle + \sum_{j \in \mathcal{J}} a_j^{obj} x_j + b^{obj},$$

in terms of the symmetric matrices, F_j^{obj} , and scalars, a_j^{obj} and b^{obj} .

- The affine expressions of the scalar constraints are defined, for $i \in \mathcal{I}$, as

$$g_i = \sum_{j \in \mathcal{J}^{PSD}} \langle F_{ij}, X_j \rangle + \sum_{j \in \mathcal{J}} a_{ij} x_j + b_i,$$

in terms of the symmetric matrices, F_{ij} , and scalars, a_{ij} and b_i .

- The affine expressions of the PSD constraints are defined, for $i \in \mathcal{I}^{PSD}$, as

$$G_i = \sum_{j \in \mathcal{J}} x_j H_{ij} + D_i,$$

in terms of the symmetric matrices, H_{ij} and D_i .

List of cones

The format uses an explicit syntax for symmetric positive semidefinite cones as shown above. For scalar variables and constraints, constructed in vectors, the supported conic domains and their minimum sizes are given as follows.

Table 15.3: Cones available in the CBF format

Name	CBF keyword	Cone family
Free domain	F	linear
Positive orthant	L+	linear
Negative orthant	L-	linear
Fixpoint zero	L=	linear
Quadratic cone	Q	second-order
Rotated quadratic cone	QR	second-order

15.4.4 File Format Keywords

VER

Description: The version of the Conic Benchmark Format used to write the file.

HEADER: None

BODY: One line formatted as:

INT

This is the version number.

Must appear exactly once in a file, as the first keyword.

OBJSENSE

Description: Define the objective sense.

HEADER: None

BODY: One line formatted as:

STR

having MIN indicates minimize, and MAX indicates maximize. Capital letters are required.

Must appear exactly once in a file.

PSDVAR

Description: Construct the PSD variables.

HEADER: One line formatted as:

INT

This is the number of PSD variables in the problem.

BODY: A list of lines formatted as:

INT

This indicates the number of rows (equal to the number of columns) in the matrix-valued PSD variable. The number of lines should match the number stated in the header.

VAR

Description: Construct the scalar variables.

HEADER: One line formatted as:

INT INT

This is the number of scalar variables, followed by the number of conic domains they are restricted to.

BODY: A list of lines formatted as:

STR INT

This indicates the cone name (see [Table 15.3](#)), and the number of scalar variables restricted to this cone. These numbers should add up to the number of scalar variables stated first in the header. The number of lines should match the second number stated in the header.

INT

Description: Declare integer requirements on a selected subset of scalar variables.

HEADER: one line formatted as:

INT

This is the number of integer scalar variables in the problem.

BODY: a list of lines formatted as:

INT

This indicates the scalar variable index $j \in \mathcal{J}$. The number of lines should match the number stated in the header.

Can only be used after the keyword **VAR**.

PSDCON

Description: Construct the PSD constraints.

HEADER: One line formatted as:

INT

This is the number of PSD constraints in the problem.

BODY: A list of lines formatted as:

INT

This indicates the number of rows (equal to the number of columns) in the matrix-valued affine expression of the PSD constraint. The number of lines should match the number stated in the header.

Can only be used after these keywords: **PSDVAR**, **VAR**.

CON

Description: Construct the scalar constraints.

HEADER: One line formatted as:

INT INT

This is the number of scalar constraints, followed by the number of conic domains they restrict to.

BODY: A list of lines formatted as:

STR INT

This indicates the cone name (see [Table 15.3](#)), and the number of affine expressions restricted to this cone. These numbers should add up to the number of scalar constraints stated first in the header. The number of lines should match the second number stated in the header.

Can only be used after these keywords: **PSDVAR**, **VAR**

OBJFCOORD

Description: Input sparse coordinates (quadruplets) to define the symmetric matrices F_j^{obj} , as used in the objective.

HEADER: One line formatted as:

INT

This is the number of coordinates to be specified.
BODY: A list of lines formatted as:

INT INT INT REAL

This indicates the PSD variable index $j \in \mathcal{J}^{PSD}$, the row index, the column index and the coefficient value. The number of lines should match the number stated in the header.

OBJACOORD

Description: Input sparse coordinates (pairs) to define the scalars, a_j^{obj} , as used in the objective.

HEADER: One line formatted as:

INT

This is the number of coordinates to be specified.
BODY: A list of lines formatted as:

INT REAL

This indicates the scalar variable index $j \in \mathcal{J}$ and the coefficient value. The number of lines should match the number stated in the header.

OBJBCOORD

Description: Input the scalar, b^{obj} , as used in the objective.

HEADER: None.

BODY: One line formatted as:

REAL

This indicates the coefficient value.

FCOORD

Description: Input sparse coordinates (quintuplets) to define the symmetric matrices, F_{ij} , as used in the scalar constraints.

HEADER: One line formatted as:

INT

This is the number of coordinates to be specified.
BODY: A list of lines formatted as:

INT INT INT INT REAL

This indicates the scalar constraint index $i \in \mathcal{I}$, the PSD variable index $j \in \mathcal{J}^{PSD}$, the row index, the column index and the coefficient value. The number of lines should match the number stated in the header.

ACOORD

Description: Input sparse coordinates (triplets) to define the scalars, a_{ij} , as used in the scalar constraints.

HEADER: One line formatted as:

INT

This is the number of coordinates to be specified.
BODY: A list of lines formatted as:

INT INT REAL

This indicates the scalar constraint index $i \in \mathcal{I}$, the scalar variable index $j \in \mathcal{J}$ and the coefficient value. The number of lines should match the number stated in the header.

BCOORD

Description: Input sparse coordinates (pairs) to define the scalars, b_i , as used in the scalar constraints.

HEADER: One line formatted as:

INT

This is the number of coordinates to be specified.

BODY: A list of lines formatted as:

INT REAL

This indicates the scalar constraint index $i \in \mathcal{I}$ and the coefficient value. The number of lines should match the number stated in the header.

HCOORD

Description: Input sparse coordinates (quintuplets) to define the symmetric matrices, H_{ij} , as used in the PSD constraints.

HEADER: One line formatted as:

INT

This is the number of coordinates to be specified.

BODY: A list of lines formatted as

INT INT INT INT REAL

This indicates the PSD constraint index $i \in \mathcal{I}^{PSD}$, the scalar variable index $j \in \mathcal{J}$, the row index, the column index and the coefficient value. The number of lines should match the number stated in the header.

DCOORD

Description: Input sparse coordinates (quadruplets) to define the symmetric matrices, D_i , as used in the PSD constraints.

HEADER: One line formatted as

INT

This is the number of coordinates to be specified.

BODY: A list of lines formatted as:

INT INT INT REAL

This indicates the PSD constraint index $i \in \mathcal{I}^{PSD}$, the row index, the column index and the coefficient value. The number of lines should match the number stated in the header.

CHANGE

Start of a new instance specification based on changes to the previous. Can be interpreted as the end of file when the hotstart-sequence is unsupported or undesired.

BODY: None

Header: None

15.4.5 CBF Format Examples

Minimal Working Example

The conic optimization problem (15.11), has three variables in a quadratic cone - first one is integer - and an affine expression in domain 0 (equality constraint).

$$\begin{aligned} & \text{minimize} && 5.1 x_0 \\ & \text{subject to} && 6.2 x_1 + 7.3 x_2 - 8.4 \in \{0\} \\ & && x \in \mathcal{Q}^3, x_0 \in \mathbb{Z}. \end{aligned} \tag{15.11}$$

Its formulation in the Conic Benchmark Format begins with the version of the CBF format used, to safeguard against later revisions.

```
VER
1
```

Next follows the problem structure, consisting of the objective sense, the number and domain of variables, the indices of integer variables, and the number and domain of scalar-valued affine expressions (i.e., the equality constraint).

```
OBJSENSE
MIN

VAR
3 1
Q 3

INT
1
0

CON
1 1
L= 1
```

Finally follows the problem data, consisting of the coefficients of the objective, the coefficients of the constraints, and the constant terms of the constraints. All data is specified on a sparse coordinate form.

```
OBJCOORD
1
0 5.1

ACCOORD
2
0 1 6.2
0 2 7.3

BCCOORD
1
0 -8.4
```

This concludes the example.

Mixing Linear, Second-order and Semidefinite Cones

The conic optimization problem (15.12), has a semidefinite cone, a quadratic cone over unordered subindices, and two equality constraints.

$$\begin{aligned}
 & \text{minimize} && \left\langle \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}, X_1 \right\rangle + x_1 \\
 & \text{subject to} && \left\langle \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, X_1 \right\rangle + x_1 &= 1.0, \\
 & && \left\langle \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, X_1 \right\rangle + x_0 + x_2 &= 0.5, \\
 & && x_1 \geq \sqrt{x_0^2 + x_2^2}, \\
 & && X_1 \succeq \mathbf{0}.
 \end{aligned} \tag{15.12}$$

The equality constraints are easily rewritten to the conic form, $(g_0, g_1) \in \{0\}^2$, by moving constants such that the right-hand-side becomes zero. The quadratic cone does not fit under the **VAR** keyword in this variable permutation. Instead, it takes a scalar constraint $(g_2, g_3, g_4) = (x_1, x_0, x_2) \in \mathcal{Q}^3$, with scalar

variables constructed as $(x_0, x_1, x_2) \in \mathbb{R}^3$. Its formulation in the CBF format is reported in the following list

```
# File written using this version of the Conic Benchmark Format:
#   | Version 1.
VER
1

# The sense of the objective is:
#   | Minimize.
OBJSENSE
MIN

# One PSD variable of this size:
#   | Three times three.
PSDVAR
1
3

# Three scalar variables in this one conic domain:
#   | Three are free.
VAR
3 1
F 3

# Five scalar constraints with affine expressions in two conic domains:
#   | Two are fixed to zero.
#   | Three are in conic quadratic domain.
CON
5 2
L= 2
Q 3

# Five coordinates in F^{obj}_j coefficients:
#   | F^{obj}[0][0,0] = 2.0
#   | F^{obj}[0][1,0] = 1.0
#   | and more...
OBJFCOORD
5
0 0 0 2.0
0 1 0 1.0
0 1 1 2.0
0 2 1 1.0
0 2 2 2.0

# One coordinate in a^{obj}_j coefficients:
#   | a^{obj}[1] = 1.0
OBJACOORD
1
1 1.0

# Nine coordinates in F_{ij} coefficients:
#   | F[0,0][0,0] = 1.0
#   | F[0,0][1,1] = 1.0
#   | and more...
FCOORD
9
0 0 0 0 1.0
0 0 1 1 1.0
0 0 2 2 1.0
1 0 0 0 1.0
1 0 1 0 1.0
```

(continues on next page)

(continued from previous page)

```
1 0 2 0 1.0
1 0 1 1 1.0
1 0 2 1 1.0
1 0 2 2 1.0

# Six coordinates in a_ij coefficients:
#     | a[0,1] = 1.0
#     | a[1,0] = 1.0
#     | and more...
ACCOORD
6
0 1 1.0
1 0 1.0
1 2 1.0
2 1 1.0
3 0 1.0
4 2 1.0

# Two coordinates in b_i coefficients:
#     | b[0] = -1.0
#     | b[1] = -0.5
BCCOORD
2
0 -1.0
1 -0.5
```

Mixing Semidefinite Variables and Linear Matrix Inequalities

The standard forms in semidefinite optimization are usually based either on semidefinite variables or linear matrix inequalities. In the CBF format, both forms are supported and can even be mixed as shown in.

$$\begin{aligned} & \text{minimize} && \left\langle \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, X_1 \right\rangle + x_1 + x_2 + 1 \\ & \text{subject to} && \left\langle \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, X_1 \right\rangle - x_1 - x_2 \geq 0.0, \\ & && x_1 \begin{bmatrix} 0 & 1 \\ 1 & 3 \end{bmatrix} + x_2 \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \succeq \mathbf{0}, \\ & && X_1 \succeq \mathbf{0}. \end{aligned} \tag{15.13}$$

Its formulation in the CBF format is written in what follows

```
# File written using this version of the Conic Benchmark Format:
#     | Version 1.
VER
1

# The sense of the objective is:
#     | Minimize.
OBJSENSE
MIN

# One PSD variable of this size:
#     | Two times two.
PSDVAR
1
2

# Two scalar variables in this one conic domain:
#     | Two are free.
```

(continues on next page)

```

VAR
2 1
F 2

# One PSD constraint of this size:
#   | Two times two.
PSDCON
1
2

# One scalar constraint with an affine expression in this one conic domain:
#   | One is greater than or equal to zero.
CON
1 1
L+ 1

# Two coordinates in  $F^{\text{obj}}_j$  coefficients:
#   |  $F^{\text{obj}}[0][0,0] = 1.0$ 
#   |  $F^{\text{obj}}[0][1,1] = 1.0$ 
OBJFCOORD
2
0 0 0 1.0
0 1 1 1.0

# Two coordinates in  $a^{\text{obj}}_j$  coefficients:
#   |  $a^{\text{obj}}[0] = 1.0$ 
#   |  $a^{\text{obj}}[1] = 1.0$ 
OBJACOORD
2
0 1.0
1 1.0

# One coordinate in  $b^{\text{obj}}$  coefficient:
#   |  $b^{\text{obj}} = 1.0$ 
OBJBCOORD
1.0

# One coordinate in  $F_{ij}$  coefficients:
#   |  $F[0,0][1,0] = 1.0$ 
FCOORD
1
0 0 1 0 1.0

# Two coordinates in  $a_{ij}$  coefficients:
#   |  $a[0,0] = -1.0$ 
#   |  $a[0,1] = -1.0$ 
ACOORD
2
0 0 -1.0
0 1 -1.0

# Four coordinates in  $H_{ij}$  coefficients:
#   |  $H[0,0][1,0] = 1.0$ 
#   |  $H[0,0][1,1] = 3.0$ 
#   | and more...
HCOORD
4
0 0 1 0 1.0
0 0 1 1 3.0
0 1 0 0 3.0
0 1 1 0 1.0

```

```
# Two coordinates in D_i coefficients:
#   | D[0][0,0] = -1.0
#   | D[0][1,1] = -1.0
DCOORD
2
0 0 0 -1.0
0 1 1 -1.0
```

Optimization Over a Sequence of Objectives

The linear optimization problem (15.14), is defined for a sequence of objectives such that hotstarting from one to the next might be advantages.

$$\begin{aligned} & \text{maximize}_k && g_k^{obj} \\ & \text{subject to} && 50x_0 + 31 \leq 250, \\ & && 3x_0 - 2x_1 \geq -4, \\ & && x \in \mathbb{R}_+^2, \end{aligned} \tag{15.14}$$

given,

1. $g_0^{obj} = x_0 + 0.64x_1$.
2. $g_1^{obj} = 1.11x_0 + 0.76x_1$.
3. $g_2^{obj} = 1.11x_0 + 0.85x_1$.

Its formulation in the CBF format is reported in Listing 15.5.

Listing 15.5: Problem (15.14) in CBF format.

```
# File written using this version of the Conic Benchmark Format:
#   | Version 1.
VER
1

# The sense of the objective is:
#   | Maximize.
OBJSENSE
MAX

# Two scalar variables in this one conic domain:
#   | Two are nonnegative.
VAR
2 1
L+ 2

# Two scalar constraints with affine expressions in these two conic domains:
#   | One is in the nonpositive domain.
#   | One is in the nonnegative domain.
CON
2 2
L- 1
L+ 1

# Two coordinates in a^{obj}_j coefficients:
#   | a^{obj}[0] = 1.0
#   | a^{obj}[1] = 0.64
OBJACCOORD
2
0 1.0
```

(continues on next page)

```

1 0.64

# Four coordinates in a_ij coefficients:
#   | a[0,0] = 50.0
#   | a[1,0] = 3.0
#   | and more...
ACCOORD
4
0 0 50.0
1 0 3.0
0 1 31.0
1 1 -2.0

# Two coordinates in b_i coefficients:
#   | b[0] = -250.0
#   | b[1] = 4.0
BCCOORD
2
0 -250.0
1 4.0

# New problem instance defined in terms of changes.
CHANGE

# Two coordinate changes in a^{obj}_j coefficients. Now it is:
#   | a^{obj}[0] = 1.11
#   | a^{obj}[1] = 0.76
OBJACCOORD
2
0 1.11
1 0.76

# New problem instance defined in terms of changes.
CHANGE

# One coordinate change in a^{obj}_j coefficients. Now it is:
#   | a^{obj}[0] = 1.11
#   | a^{obj}[1] = 0.85
OBJACCOORD
1
1 0.85

```

15.5 The PTF Format

The PTF format is a new human-readable, natural text format. Its features and structure are similar to the *OPF* format, with the difference that the PTF format **does** support semidefinite terms.

15.5.1 The overall format

The format is indentation based, where each section is started by a head line and followed by a section body with deeper indentation than the head line. For example:

```

Header line
  Body line 1
  Body line 1
  Body line 1

```

Section can also be nested:

```
Header line A
  Body line in A
    Header line A.1
      Body line in A.1
      Body line in A.1
    Body line in A
```

The indentation of blank lines is ignored, so a subsection can contain a blank line with no indentation. The character # defines a line comment and anything between the # character and the end of the line is ignored.

In a PTF file, the first section must be a **Task** section. The order of the remaining section is arbitrary, and sections may occur multiple times or not at all.

MOSEK will ignore any top-level section it does not recognize.

Names

In the description of the format we use following definitions for name strings:

```
NAME: PLAIN_NAME | QUOTED_NAME
PLAIN_NAME: [a-zA-Z_] [a-zA-Z0-9_-.!|]
QUOTED_NAME: '"' ( [^'\\r\n] | "\\\" ( [\\rn] | \"x\" [0-9a-fA-F] [0-9a-fA-F] ) ) * '"'
```

Expressions

An expression is a sum of terms. A term is either a linear term (a coefficient and a variable name, where the coefficient can be left out if it is 1.0), or a matrix inner product.

An expression:

```
EXPR: EMPTY | [+]? TERM ( [+]? TERM ) *
TERM: LINEAR_TERM | MATRIX_TERM
```

A linear term

```
LINEAR_TERM: FLOAT? NAME
```

A matrix term

```
MATRIX_TERM: "<" FLOAT? NAME ( [+]? FLOAT? NAME ) * ";" NAME ">"
```

Here the right-hand name is the name of a (semidefinite) matrix variable, and the left-hand side is a sum of symmetric matrixes. The actual matrixes are defined in a separate section.

Expressions can span multiple lines by giving subsequent lines a deeper indentation.

For example following two section are equivalent:

```
# Everything on one line:
x1 + x2 + x3 + x4

# Split into multiple lines:
x1
  + x2
  + x3
  + x4
```

15.5.2 Task section

The first section of the file must be a **Task**. The text in this section is not used and may contain comments, or meta-information from the writer or about the content.

Format:

```
Task NAME
  Anything goes here...
```

NAME is a the task name.

15.5.3 Objective section

The **Objective** section defines the objective name, sense and function. The format:

```
"Objective" NAME?  
  ( "Minimize" | "Maximize" ) EXPR
```

For example:

```
Objective 'obj'  
Minimize x1 + 0.2 x2 + < M1 ; X1 >
```

15.5.4 Constraints section

The constraints section defines a series of constraints. A constraint defines a term $A \cdot x + b \in K$. For linear constraints A is just one row, while for conic constraints it can be multiple rows. If a constraint spans multiple rows these can either be written inline separated by semi-colons, or each expression in a separate sub-section.

Simple linear constraints:

```
"Constraints"  
NAME? "[" [-+] (FLOAT | "Inf") (";" [-+] (FLOAT | "Inf"))? "]" EXPR
```

If the brackets contain two values, they are used as upper and lower bounds. If they contain one value the constraint is an equality.

For example:

```
Constraints  
'c1' [0;10] x1 + x2 + x3  
[0] x1 + x2 + x3
```

Constraint blocks put the expression either in a subsection or inline. The cone type (domain) is written in the brackets, and **MOSEK** currently supports following types:

- SOC(N) Second order cone of dimension N
- RSOC(N) Rotated second order cone of dimension N
- PSD(N) Symmetric positive semidefinite cone of dimension N. This contains $N \cdot (N+1) / 2$ elements.
- PEXP Primal exponential cone of dimension 3
- DEXP Dual exponential cone of dimension 3
- PPOW(N,P) Primal power cone of dimension N with parameter P
- DPOW(N,P) Dual power cone of dimension N with parameter P
- ZERO(N) The zero-cone of dimension N.

```
"Constraints"  
NAME? "[" DOMAIN "]" EXPR_LIST
```

For example:

```
Constraints  
'K1' [SOC(3)] x1 + x2 ; x2 + x3 ; x3 + x1  
'K2' [RSOC(3)]  
  x1 + x2  
  x2 + x3  
  x3 + x1
```

15.5.5 Variables section

Any variable used in an expression must be defined in a variable section. The variable section defines each variable domain.

```
"Variables"
  NAME "[" [-+] (FLOAT | "Inf") (";" [-+] (FLOAT | "Inf") )? "]"
  NAME "[" DOMAIN "]" NAMES

  For example, a linear variable
```

```
Variables
  x1 [0;Inf]
```

As with constraints, members of a conic domain can be listed either inline or in a subsection:

```
Variables
  k1 [SOC(3)] x1 ; x2 ; x3
  k2 [RSOC(3)]
    x1
    x2
    x3
```

15.5.6 Integer section

This section contains a list of variables that are integral. For example:

```
Integer
  x1 x2 x3
```

15.5.7 SymmetricMatrixes section

This section defines the symmetric matrixes used for matrix coefficients in matrix inner product terms. The section lists named matrixes, each with a size and a number of non-zeros. Only non-zeros in the lower triangular part should be defined.

```
"SymmetricMatrixes"
  NAME "SYMMAT" "(" INT ")" ( "(" INT "," INT "," FLOAT ")" ) *
  ...
```

For example:

```
SymmetricMatrixes
  M1 SYMMAT(3) (0,0,1.0) (1,1,2.0) (2,1,0.5)
  M2 SYMMAT(3)
    (0,0,1.0)
    (1,1,2.0)
    (2,1,0.5)
```

15.5.8 Solutions section

Each subsection defines a solution. A solution defines for each constraint and for each variable exactly one primal value and either one (for conic domains) or two (for linear domains) dual values. The values follow the same logic as in the **MOSEK C API**. A primal and a dual solution status defines the meaning of the values primal and dual (solution, certificate, unknown, etc.)

The format is this:

```
"Solutions"
  "Solution" WHICH SOL
    "ProblemStatus" PROSTA PROSTA?
    "SolutionStatus" SOLSTA SOLSTA?
```

(continues on next page)

```

"Objective" FLOAT FLOAT
"Variables"
  # Linear variable status: level, slx, sux
  NAME "[" STATUS "]" FLOAT (FLOAT FLOAT)?
  # Conic variable status: level, snx
  NAME
    "[" STATUS "]" FLOAT FLOAT?
  ...
"Constraints"
  # Linear variable status: level, slx, sux
  NAME "[" STATUS "]" FLOAT (FLOAT FLOAT)?
  # Conic variable status: level, snx
  NAME
    "[" STATUS "]" FLOAT FLOAT?
  ...

```

Following values for **WHICHSOL** are supported:

- **interior** Interior solution, the result of an interior-point solver.
- **basic** Basic solution, as produced by a simplex solver.
- **integer** Integer solution, the solution to a mixed-integer problem. This does not define a dual solution.

Following values for **PROSTA** are supported:

- **unknown** The problem status is unknown
- **feasible** The problem has been proven feasible
- **infeasible** The problem has been proven infeasible
- **illposed** The problem has been proved to be ill posed
- **infeasible_or_unbounded** The problem is infeasible or unbounded

Following values for **SOLSTA** are supported:

- **unknown** The solution status is unknown
- **feasible** The solution is feasible
- **optimal** The solution is optimal
- **infeas_cert** The solution is a certificate of infeasibility
- **illposed_cert** The solution is a certificate of illposedness

Following values for **STATUS** are supported:

- **unknown** The value is unknown
- **super_basic** The value is super basic
- **at_lower** The value is basic and at its lower bound
- **at_upper** The value is basic and at its upper bound
- **fixed** The value is basic fixed
- **infinite** The value is at infinity

15.6 The Task Format

The Task format is **MOSEK**'s native binary format. It contains a complete image of a **MOSEK** task, i.e.

- Problem data: Linear, conic, semidefinite and quadratic data
- Problem item names: Variable names, constraints names, cone names etc.
- Parameter settings
- Solutions

There are a few things to be aware of:

- Status of a solution read from a file will *always* be unknown.
- Parameter settings in a task file *always override* any parameters set on the command line or in a parameter file.

The format is based on the *TAR* (USTar) file format. This means that the individual pieces of data in a `.task` file can be examined by unpacking it as a *TAR* file. Please note that the inverse may not work: Creating a file using *TAR* will most probably not create a valid **MOSEK** Task file since the order of the entries is important.

15.7 The JSON Format

MOSEK provides the possibility to read/write problems in valid JSON format.

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

The official JSON website <http://www.json.org> provides plenty of information along with the format definition.

MOSEK defines two JSON-like formats:

- `jtask`
- `jsol`

Despite being text-based human-readable formats, *jtask* and *jsol* files will include no indentation and no new-lines, in order to keep the files as compact as possible. We therefore strongly advise to use JSON viewer tools to inspect *jtask* and *jsol* files.

15.7.1 *jtask* format

It stores a problem instance. The *jtask* format contains the same information as a *task format*. Even though a *jtask* file is human-readable, we do not recommend users to create it by hand, but to rely on **MOSEK**.

15.7.2 *jsol* format

It stores a problem solution. The *jsol* format contains all solutions and information items.

15.7.3 A *jtask* example

In Listing 15.6 we present a file in the *jtask* format that corresponds to the sample problem from `lo1.lp`. The listing has been formatted for readability.

Listing 15.6: A formatted *jtask* file for the `lo1.lp` example.

```
{
  "$schema": "http://mosek.com/json/schema#",
  "Task/INFO": {
    "taskname": "lo1",
    "numvar": 4,
    "numcon": 3,
    "numcone": 0,
    "numbarvar": 0,
    "numanz": 9,
    "numsymmat": 0,
    "mosekver": [
      8,
      0,
      0,
      9
    ]
  },
  "Task/data": {
    "var": {
      "name": [
        "x1",
        "x2",
        "x3",
        "x4"
      ],
      "bk": [
        "lo",
        "ra",
        "lo",
        "lo"
      ],
      "b1": [
        0.0,
        0.0,
        0.0,
        0.0
      ],
      "bu": [
        1e+30,
        1e+1,
        1e+30,
        1e+30
      ],
      "type": [
        "cont",
        "cont",
        "cont",
        "cont"
      ]
    },
    "con": {
      "name": [
        "c1",
        "c2",
        "c3"
      ],

```

(continues on next page)

```

        "bk": [
            "fx",
            "lo",
            "up"
        ],
        "b1": [
            3e+1,
            1.5e+1,
            -1e+30
        ],
        "bu": [
            3e+1,
            1e+30,
            2.5e+1
        ]
    },
    "objective": {
        "sense": "max",
        "name": "obj",
        "c": {
            "subj": [
                0,
                1,
                2,
                3
            ],
            "val": [
                3e+0,
                1e+0,
                5e+0,
                1e+0
            ]
        },
        "cfix": 0.0
    },
    "A": {
        "subi": [
            0,
            0,
            0,
            1,
            1,
            1,
            1,
            1,
            2,
            2
        ],
        "subj": [
            0,
            1,
            2,
            0,
            1,
            2,
            3,
            1,
            3
        ],
        "val": [
            3e+0,
            1e+0,

```

```

        2e+0,
        2e+0,
        1e+0,
        3e+0,
        1e+0,
        2e+0,
        3e+0
    ]
}
},
"Task/parameters":{
    "iparam":{
        "ANA_SOL_BASIS":"ON",
        "ANA_SOL_PRINT_VIOLATED":"OFF",
        "AUTO_SORT_A_BEFORE_OPT":"OFF",
        "AUTO_UPDATE_SOL_INFO":"OFF",
        "BASIS_SOLVE_USE_PLUS_ONE":"OFF",
        "BI_CLEAN_OPTIMIZER":"OPTIMIZER_FREE",
        "BI_IGNORE_MAX_ITER":"OFF",
        "BI_IGNORE_NUM_ERROR":"OFF",
        "BI_MAX_ITERATIONS":1000000,
        "CACHE_LICENSE":"ON",
        "CHECK_CONVEXITY":"CHECK_CONVEXITY_FULL",
        "COMPRESS_STATFILE":"ON",
        "CONCURRENT_NUM_OPTIMIZERS":2,
        "CONCURRENT_PRIORITY_DUAL_SIMPLEX":2,
        "CONCURRENT_PRIORITY_FREE_SIMPLEX":3,
        "CONCURRENT_PRIORITY_INTPNT":4,
        "CONCURRENT_PRIORITY_PRIMAL_SIMPLEX":1,
        "FEASREPAIR_OPTIMIZE":"FEASREPAIR_OPTIMIZE_NONE",
        "INFEAS_GENERIC_NAMES":"OFF",
        "INFEAS_PREFER_PRIMAL":"ON",
        "INFEAS_REPORT_AUTO":"OFF",
        "INFEAS_REPORT_LEVEL":1,
        "INTPNT_BASIS":"BI_ALWAYS",
        "INTPNT_DIFF_STEP":"ON",
        "INTPNT_FACTOR_DEBUG_LVL":0,
        "INTPNT_FACTOR_METHOD":0,
        "INTPNT_HOTSTART":"INTPNT_HOTSTART_NONE",
        "INTPNT_MAX_ITERATIONS":400,
        "INTPNT_MAX_NUM_COR":-1,
        "INTPNT_MAX_NUM_REFINEMENT_STEPS":-1,
        "INTPNT_OFF_COL_TRH":40,
        "INTPNT_ORDER_METHOD":"ORDER_METHOD_FREE",
        "INTPNT_REGULARIZATION_USE":"ON",
        "INTPNT_SCALING":"SCALING_FREE",
        "INTPNT_SOLVE_FORM":"SOLVE_FREE",
        "INTPNT_STARTING_POINT":"STARTING_POINT_FREE",
        "LIC_TRH_EXPIRY_WRN":7,
        "LICENSE_DEBUG":"OFF",
        "LICENSE_PAUSE_TIME":0,
        "LICENSE_SUPPRESS_EXPIRE_WRNS":"OFF",
        "LICENSE_WAIT":"OFF",
        "LOG":10,
        "LOG_ANA_PRO":1,
        "LOG_BI":4,
        "LOG_BI_FREQ":2500,
        "LOG_CHECK_CONVEXITY":0,
        "LOG_CONCURRENT":1,
        "LOG_CUT_SECOND_OPT":1,
        "LOG_EXPAND":0,

```

```

"LOG_FACTOR":1,
"LOG_FEAS_REPAIR":1,
"LOG_FILE":1,
"LOG_HEAD":1,
"LOG_INFEAS_ANA":1,
"LOG_INTPNT":4,
"LOG_MIO":4,
"LOG_MIO_FREQ":1000,
"LOG_OPTIMIZER":1,
"LOG_ORDER":1,
"LOG_PRESOLVE":1,
"LOG_RESPONSE":0,
"LOG_SENSITIVITY":1,
"LOG_SENSITIVITY_OPT":0,
"LOG_SIM":4,
"LOG_SIM_FREQ":1000,
"LOG_SIM_MINOR":1,
"LOG_STORAGE":1,
"MAX_NUM_WARNINGS":10,
"MIO_BRANCH_DIR":"BRANCH_DIR_FREE",
"MIO_CONSTRUCT_SOL":"OFF",
"MIO_CUT_CLIQUE":"ON",
"MIO_CUT_CMIR":"ON",
"MIO_CUT_GMI":"ON",
"MIO_CUT_KNAPSACK_COVER":"OFF",
"MIO_HEURISTIC_LEVEL":-1,
"MIO_MAX_NUM_BRANCHES":-1,
"MIO_MAX_NUM_RELAXS":-1,
"MIO_MAX_NUM_SOLUTIONS":-1,
"MIO_MODE":"MIO_MODE_SATISFIED",
"MIO_MT_USER_CB":"ON",
"MIO_NODE_OPTIMIZER":"OPTIMIZER_FREE",
"MIO_NODE_SELECTION":"MIO_NODE_SELECTION_FREE",
"MIO_PERSPECTIVE_REFORMULATE":"ON",
"MIO_PROBING_LEVEL":-1,
"MIO_RINS_MAX_NODES":-1,
"MIO_ROOT_OPTIMIZER":"OPTIMIZER_FREE",
"MIO_ROOT_REPEAT_PRESOLVE_LEVEL":-1,
"MT_SPINCOUNT":0,
"NUM_THREADS":0,
"OPF_MAX_TERMS_PER_LINE":5,
"OPF_WRITE_HEADER":"ON",
"OPF_WRITE_HINTS":"ON",
"OPF_WRITE_PARAMETERS":"OFF",
"OPF_WRITE_PROBLEM":"ON",
"OPF_WRITE_SOL_BAS":"ON",
"OPF_WRITE_SOL_ITG":"ON",
"OPF_WRITE_SOL_ITR":"ON",
"OPF_WRITE_SOLUTIONS":"OFF",
"OPTIMIZER":"OPTIMIZER_FREE",
"PARAM_READ_CASE_NAME":"ON",
"PARAM_READ_IGN_ERROR":"OFF",
"PRESOLVE_ELIMINATOR_MAX_FILL":-1,
"PRESOLVE_ELIMINATOR_MAX_NUM_TRIES":-1,
"PRESOLVE_LEVEL":-1,
"PRESOLVE_LINDEP_ABS_WORK_TRH":100,
"PRESOLVE_LINDEP_REL_WORK_TRH":100,
"PRESOLVE_LINDEP_USE":"ON",
"PRESOLVE_MAX_NUM_REDUCATIONS":-1,
"PRESOLVE_USE":"PRESOLVE_MODE_FREE",
"PRIMAL_REPAIR_OPTIMIZER":"OPTIMIZER_FREE",

```

```

"QO_SEPARABLE_REFORMULATION": "OFF",
"READ_DATA_COMPRESSED": "COMPRESS_FREE",
"READ_DATA_FORMAT": "DATA_FORMAT_EXTENSION",
"READ_DEBUG": "OFF",
"READ_KEEP_FREE_CON": "OFF",
"READ_LP_DROP_NEW_VARS_IN_BOU": "OFF",
"READ_LP_QUOTED_NAMES": "ON",
"READ_MPS_FORMAT": "MPS_FORMAT_FREE",
"READ_MPS_WIDTH": 1024,
"READ_TASK_IGNORE_PARAM": "OFF",
"SENSITIVITY_ALL": "OFF",
"SENSITIVITY_OPTIMIZER": "OPTIMIZER_FREE_SIMPLEX",
"SENSITIVITY_TYPE": "SENSITIVITY_TYPE_BASIS",
"SIM_BASIS_FACTOR_USE": "ON",
"SIM_DEGEN": "SIM_DEGEN_FREE",
"SIM_DUAL_CRASH": 90,
"SIM_DUAL_PHASEONE_METHOD": 0,
"SIM_DUAL_RESTRICT_SELECTION": 50,
"SIM_DUAL_SELECTION": "SIM_SELECTION_FREE",
"SIM_EXPLOIT_DUPVEC": "SIM_EXPLOIT_DUPVEC_OFF",
"SIM_HOTSTART": "SIM_HOTSTART_FREE",
"SIM_HOTSTART_LU": "ON",
"SIM_INTEGER": 0,
"SIM_MAX_ITERATIONS": 10000000,
"SIM_MAX_NUM_SETBACKS": 250,
"SIM_NON_SINGULAR": "ON",
"SIM_PRIMAL_CRASH": 90,
"SIM_PRIMAL_PHASEONE_METHOD": 0,
"SIM_PRIMAL_RESTRICT_SELECTION": 50,
"SIM_PRIMAL_SELECTION": "SIM_SELECTION_FREE",
"SIM_REFACTOR_FREQ": 0,
"SIM_REFORMULATION": "SIM_REFORMULATION_OFF",
"SIM_SAVE_LU": "OFF",
"SIM_SCALING": "SCALING_FREE",
"SIM_SCALING_METHOD": "SCALING_METHOD_POW2",
"SIM_SOLVE_FORM": "SOLVE_FREE",
"SIM_STABILITY_PRIORITY": 50,
"SIM_SWITCH_OPTIMIZER": "OFF",
"SOL_FILTER_KEEP_BASIC": "OFF",
"SOL_FILTER_KEEP_RANGED": "OFF",
"SOL_READ_NAME_WIDTH": -1,
"SOL_READ_WIDTH": 1024,
"SOLUTION_CALLBACK": "OFF",
"TIMING_LEVEL": 1,
"WRITE_BAS_CONSTRAINTS": "ON",
"WRITE_BAS_HEAD": "ON",
"WRITE_BAS_VARIABLES": "ON",
"WRITE_DATA_COMPRESSED": 0,
"WRITE_DATA_FORMAT": "DATA_FORMAT_EXTENSION",
"WRITE_DATA_PARAM": "OFF",
"WRITE_FREE_CON": "OFF",
"WRITE_GENERIC_NAMES": "OFF",
"WRITE_GENERIC_NAMES_IO": 1,
"WRITE_IGNORE_INCOMPATIBLE_CONIC_ITEMS": "OFF",
"WRITE_IGNORE_INCOMPATIBLE_ITEMS": "OFF",
"WRITE_IGNORE_INCOMPATIBLE_NL_ITEMS": "OFF",
"WRITE_IGNORE_INCOMPATIBLE_PSD_ITEMS": "OFF",
"WRITE_INT_CONSTRAINTS": "ON",
"WRITE_INT_HEAD": "ON",
"WRITE_INT_VARIABLES": "ON",
"WRITE_LP_FULL_OBJ": "ON",

```

```

"WRITE_LP_LINE_WIDTH":80,
"WRITE_LP_QUOTED_NAMES":"ON",
"WRITE_LP_STRICT_FORMAT":"OFF",
"WRITE_LP_TERMS_PER_LINE":10,
"WRITE_MPS_FORMAT":"MPS_FORMAT_FREE",
"WRITE_MPS_INT":"ON",
"WRITE_PRECISION":15,
"WRITE_SOL_BARVARIABLES":"ON",
"WRITE_SOL_CONSTRAINTS":"ON",
"WRITE_SOL_HEAD":"ON",
"WRITE_SOL_IGNORE_INVALID_NAMES":"OFF",
"WRITE_SOL_VARIABLES":"ON",
"WRITE_TASK_INC_SOL":"ON",
"WRITE_XML_MODE":"WRITE_XML_MODE_ROW"
},
"dparam":{
  "ANA_SOL_INFEAS_TOL":1e-6,
  "BASIS_REL_TOL_S":1e-12,
  "BASIS_TOL_S":1e-6,
  "BASIS_TOL_X":1e-6,
  "CHECK_CONVEXITY_REL_TOL":1e-10,
  "DATA_TOL_AIJ":1e-12,
  "DATA_TOL_AIJ_HUGE":1e+20,
  "DATA_TOL_AIJ_LARGE":1e+10,
  "DATA_TOL_BOUND_INF":1e+16,
  "DATA_TOL_BOUND_WRN":1e+8,
  "DATA_TOL_C_HUGE":1e+16,
  "DATA_TOL_CJ_LARGE":1e+8,
  "DATA_TOL_QIJ":1e-16,
  "DATA_TOL_X":1e-8,
  "FEASREPAIR_TOL":1e-10,
  "INTPNT_CO_TOL_DFEAS":1e-8,
  "INTPNT_CO_TOL_INFEAS":1e-10,
  "INTPNT_CO_TOL_MU_RED":1e-8,
  "INTPNT_CO_TOL_NEAR_REL":1e+3,
  "INTPNT_CO_TOL_PFEAS":1e-8,
  "INTPNT_CO_TOL_REL_GAP":1e-7,
  "INTPNT_NL_MERIT_BAL":1e-4,
  "INTPNT_NL_TOL_DFEAS":1e-8,
  "INTPNT_NL_TOL_MU_RED":1e-12,
  "INTPNT_NL_TOL_NEAR_REL":1e+3,
  "INTPNT_NL_TOL_PFEAS":1e-8,
  "INTPNT_NL_TOL_REL_GAP":1e-6,
  "INTPNT_NL_TOL_REL_STEP":9.95e-1,
  "INTPNT_QO_TOL_DFEAS":1e-8,
  "INTPNT_QO_TOL_INFEAS":1e-10,
  "INTPNT_QO_TOL_MU_RED":1e-8,
  "INTPNT_QO_TOL_NEAR_REL":1e+3,
  "INTPNT_QO_TOL_PFEAS":1e-8,
  "INTPNT_QO_TOL_REL_GAP":1e-8,
  "INTPNT_TOL_DFEAS":1e-8,
  "INTPNT_TOL_DSAFE":1e+0,
  "INTPNT_TOL_INFEAS":1e-10,
  "INTPNT_TOL_MU_RED":1e-16,
  "INTPNT_TOL_PATH":1e-8,
  "INTPNT_TOL_PFEAS":1e-8,
  "INTPNT_TOL_PSAFE":1e+0,
  "INTPNT_TOL_REL_GAP":1e-8,
  "INTPNT_TOL_REL_STEP":9.999e-1,
  "INTPNT_TOL_STEP_SIZE":1e-6,
  "LOWER_OBJ_CUT":-1e+30,

```

```

"LOWER_OBJ_CUT_FINITE_TRH":-5e+29,
"MIO_DISABLE_TERM_TIME":-1e+0,
"MIO_MAX_TIME":-1e+0,
"MIO_MAX_TIME_APRX_OPT":6e+1,
"MIO_NEAR_TOL_ABS_GAP":0.0,
"MIO_NEAR_TOL_REL_GAP":1e-3,
"MIO_REL_GAP_CONST":1e-10,
"MIO_TOL_ABS_GAP":0.0,
"MIO_TOL_ABS_RELAX_INT":1e-5,
"MIO_TOL_FEAS":1e-6,
"MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT":0.0,
"MIO_TOL_REL_GAP":1e-4,
"MIO_TOL_X":1e-6,
"OPTIMIZER_MAX_TIME":-1e+0,
"PRESOLVE_TOL_ABS_LINDEP":1e-6,
"PRESOLVE_TOL_AIJ":1e-12,
"PRESOLVE_TOL_REL_LINDEP":1e-10,
"PRESOLVE_TOL_S":1e-8,
"PRESOLVE_TOL_X":1e-8,
"QCQO_REFORMULATE_REL_DROP_TOL":1e-15,
"SEMIDEFINITE_TOL_APPROX":1e-10,
"SIM_LU_TOL_REL_PIV":1e-2,
"SIMPLEX_ABS_TOL_PIV":1e-7,
"UPPER_OBJ_CUT":1e+30,
"UPPER_OBJ_CUT_FINITE_TRH":5e+29
},
"sparam":{
  "BAS_SOL_FILE_NAME":"",
  "DATA_FILE_NAME":"examples/tools/data/lo1.mps",
  "DEBUG_FILE_NAME":"",
  "INT_SOL_FILE_NAME":"",
  "ITR_SOL_FILE_NAME":"",
  "MIO_DEBUG_STRING":"",
  "PARAM_COMMENT_SIGN":"%",
  "PARAM_READ_FILE_NAME":"",
  "PARAM_WRITE_FILE_NAME":"",
  "READ_MPS_BOU_NAME":"",
  "READ_MPS_OBJ_NAME":"",
  "READ_MPS_RAN_NAME":"",
  "READ_MPS_RHS_NAME":"",
  "SENSITIVITY_FILE_NAME":"",
  "SENSITIVITY_RES_FILE_NAME":"",
  "SOL_FILTER_XC_LOW":"",
  "SOL_FILTER_XC_UPR":"",
  "SOL_FILTER_XX_LOW":"",
  "SOL_FILTER_XX_UPR":"",
  "STAT_FILE_NAME":"",
  "STAT_KEY":"",
  "STAT_NAME":"",
  "WRITE_LP_GEN_VAR_NAME":"XMSKGEN"
}
}
}

```

15.8 The Solution File Format

MOSEK provides several solution files depending on the problem type and the optimizer used:

- *basis solution file* (extension `.bas`) if the problem is optimized using the simplex optimizer or basis identification is performed,

- *interior solution file* (extension `.sol`) if a problem is optimized using the interior-point optimizer and no basis identification is required,
- *integer solution file* (extension `.int`) if the problem contains integer constrained variables.

All solution files have the format:

NAME	: <problem name>							
PROBLEM STATUS	: <status of the problem>							
SOLUTION STATUS	: <status of the solution>							
OBJECTIVE NAME	: <name of the objective function>							
PRIMAL OBJECTIVE	: <primal objective value corresponding to the solution>							
DUAL OBJECTIVE	: <dual objective value corresponding to the solution>							
CONSTRAINTS								
INDEX	NAME	AT	ACTIVITY	LOWER LIMIT	UPPER LIMIT	DUAL LOWER	DUAL UPPER	
?	<name>	??	<a value>	<a value>	<a value>	<a value>	<a value>	
VARIABLES								
INDEX	NAME	AT	ACTIVITY	LOWER LIMIT	UPPER LIMIT	DUAL LOWER	DUAL UPPER	CONIC
↔DUAL								
?	<name>	??	<a value>	<a value>	<a value>	<a value>	<a value>	<a value>

In the example the fields ? and <> will be filled with problem and solution specific information. As can be observed a solution report consists of three sections, i.e.

- **HEADER** In this section, first the name of the problem is listed and afterwards the problem and solution status are shown. Next the primal and dual objective values are displayed.
- **CONSTRAINTS** For each constraint i of the form

$$l_i^c \leq \sum_{j=1}^n a_{ij}x_j \leq u_i^c, \quad (15.15)$$

the following information is listed:

- **INDEX**: A sequential index assigned to the constraint by **MOSEK**
- **NAME**: The name of the constraint assigned by the user.
- **AT**: The status of the constraint. In Table 15.4 the possible values of the status keys and their interpretation are shown.

Table 15.4: Status keys.

Status key	Interpretation
UN	Unknown status
BS	Is basic
SB	Is superbasic
LL	Is at the lower limit (bound)
UL	Is at the upper limit (bound)
EQ	Lower limit is identical to upper limit
**	Is infeasible i.e. the lower limit is greater than the upper limit.

- **ACTIVITY**: the quantity $\sum_{j=1}^n a_{ij}x_j^*$, where x^* is the value of the primal solution.
- **LOWER LIMIT**: the quantity l_i^c (see (15.15).)
- **UPPER LIMIT**: the quantity u_i^c (see (15.15).)
- **DUAL LOWER**: the dual multiplier corresponding to the lower limit on the constraint.
- **DUAL UPPER**: the dual multiplier corresponding to the upper limit on the constraint.
- **VARIABLES** The last section of the solution report lists information about the variables. This information has a similar interpretation as for the constraints. However, the column with the header **CONIC DUAL** is included for problems having one or more conic constraints. This column shows the dual variables corresponding to the conic constraints.

Example: lo1.sol

In Listing 15.7 we show the solution file for the lo1.opf problem.

Listing 15.7: An example of .sol file.

NAME	:				
PROBLEM STATUS	:	PRIMAL_AND_DUAL_FEASIBLE			
SOLUTION STATUS	:	OPTIMAL			
OBJECTIVE NAME	:	obj			
PRIMAL OBJECTIVE	:	8.33333333e+01			
DUAL OBJECTIVE	:	8.33333332e+01			
CONSTRAINTS					
INDEX	NAME	AT ACTIVITY	LOWER LIMIT	UPPER LIMIT	
↔DUAL LOWER		DUAL UPPER			
0	c1	EQ 3.00000000000000e+01	3.00000000e+01	3.00000000e+01	-0.
↔00000000000000e+00		-2.49999999741653e+00			
1	c2	SB 5.33333333049187e+01	1.50000000e+01	NONE	2.
↔09159033069640e-10		-0.00000000000000e+00			
2	c3	UL 2.49999999842049e+01	NONE	2.50000000e+01	-0.
↔00000000000000e+00		-3.33333332895108e-01			
VARIABLES					
INDEX	NAME	AT ACTIVITY	LOWER LIMIT	UPPER LIMIT	
↔DUAL LOWER		DUAL UPPER			
0	x1	LL 1.67020427038537e-09	0.00000000e+00	NONE	-4.
↔49999999528054e+00		-0.00000000000000e+00			
1	x2	LL 2.93510446211883e-09	0.00000000e+00	1.00000000e+01	-2.
↔16666666494915e+00		6.20868657679896e-10			
2	x3	SB 1.49999999899424e+01	0.00000000e+00	NONE	-8.
↔79123177245553e-10		-0.00000000000000e+00			
3	x4	SB 8.33333332273115e+00	0.00000000e+00	NONE	-1.
↔69795978848200e-09		-0.00000000000000e+00			

Chapter 16

List of examples

List of examples shipped in the distribution of Fusion API for C++:

Table 16.1: List of distributed examples

File	Description
TrafficNetworkModel.cc	Demonstrates a traffic network problem as a conic quadratic problem (CQO)
alan.cc	A portfolio choice model <code>alan.gms</code> from the GAMS model library
baker.cc	A small bakery revenue maximization linear problem
breaksolver.cc	Shows how to break a long-running task
callback.cc	An example of data/progress callback
ceo1.cc	A simple conic exponential problem
cqo1.cc	A simple conic quadratic problem
diet.cc	Solving Stigler's Nutrition model <code>diet</code> from the GAMS model library
duality.cc	Shows how to access the dual solution
facility_location.cc	Demonstrates a small one-facility location problem (CQO)
gp1.cc	A simple geometric program (GP) in conic form
lo1.cc	A simple linear problem
logistic.cc	Implements logistic regression and simple log-sum-exp (CEO)
lownerjohn_ellipsoids.cc	Computes the Lowner-John inner and outer ellipsoidal approximations of a polytope (SDO, Power Cone)
lpt.cc	Demonstrates how to solve the multi-processor scheduling problem and input an integer feasible point (MIP)
mico1.cc	A simple mixed-integer conic problem
milo1.cc	A simple mixed-integer linear problem
miocinitisol.cc	A simple mixed-integer linear problem with an initial guess
modelLib.cc	Library of implementations of basic functions
nearestcorr.cc	Solves the nearest correlation matrix problem (SDO, CQO)
parameters.cc	Shows how to set optimizer parameters and read information items
portfolio_1_basic.cc	Portfolio optimization - basic Markowitz model
portfolio_2_frontier.cc	Portfolio optimization - efficient frontier
portfolio_3_impact.cc	Portfolio optimization - market impact costs
portfolio_4_transaction.cc	Portfolio optimization - transaction costs
portfolio_5_cardinality.cc	Portfolio optimization - cardinality constraints
pow1.cc	A simple power cone problem
primal_svm.cc	Implements a simple soft-margin Support Vector Machine (CQO)

Continued on next page

Table 16.1 – continued from previous page

File	Description
qcqp_sdo_relaxation.cc	Demonstrate how to use SDP to solve convex relaxation of a mixed-integer QCQO problem
reoptimization.cc	Demonstrate how to modify and re-optimize a linear problem
response.cc	Demonstrates proper response handling
sdo1.cc	A simple semidefinite optimization problem
sospoly.cc	Models the cone of nonnegative polynomials and nonnegative trigonometric polynomials using Nesterov's framework
sudoku.cc	A SUDOKU solver (MIP)
total_variation.cc	Demonstrates how to solve a total variation problem (CQO)
tsp.cc	Solves a simple Travelling Salesman Problem and shows how to add constraints to a model and re-optimize (MIP)

Additional examples can be found on the **MOSEK** website and in other **MOSEK** publications.

Chapter 17

Interface changes

The section shows interface-specific changes to the **MOSEK** Fusion API for C++ in version 9.0. See the [release notes](#) for general changes and new features of the **MOSEK** Optimization Suite.

17.1 Backwards compatibility

There is a number of small changes in the *Fusion* API. Most of them should not affect standard applications of *Fusion* and very few should cause compilation errors.

- **Parameters.** Users who set parameters to tune the performance and numerical properties of the solver (termination criteria, tolerances, solving primal or dual, presolve etc.) are recommended to reevaluate such tuning. It may be that other, or default, parameter settings will be more beneficial in the current version. The hints in [Sec. 9](#) may be useful for some cases.
- Remove all *Near* problem and solution statuses i.e. `SolutionStatus.NearOptimal`, `SolutionStatus.NearCertificate`, `AccSolutionStatus.NearOptimal`, etc. See [Sec. 13.3.3](#).
- Some algebraic operators are more strict when it comes to exactly matching shapes, especially regarding shapes such as $1 \times n$, $n \times 1$ and n . The same applies to matching variable/expression and domain shapes. In some cases it may be necessary to explicitly reshape using the method `Variable.reshape` or `LinearDomain.withShape`.
- All shapes are specified with ordinary arrays instead of objects of class `Set`. The static methods of the class `Set` produce arrays as shape specifications.
- Replace `shape()` with `getShape()` and `size()` with `getSize()` in most places.
- Remove the option in `Expr.sum` to sum over a range of dimensions in a multidimensional expression.
- Remove the method `Constraint.add` and introduce `Constraint.update` (see [Sec. 7.9](#)).
- Rename `QConeDomain` to `ConeDomain`.
- `Variable` now extends `Expression`, so the number of method prototypes is reduced in some cases. A variable can be used everywhere an expression can.
- Expressions are evaluated lazily only when used in a constraint.
- **Semidefinite variables.** The preferred ways to declare semidefinite variables are:

```
M->variable(Domain::inPSDCone(3));           // A 3x3 PSD variable
M->variable(Domain::inPSDCone(3, 100));       // One hundred 3x3 PSD variables
                                              // arranged in shape [100, 3, 3]
```

17.2 Parameters

Added

- *intpntOrderGpNumSeeds*
- *logLocalInfo*
- *mioConicOuterApproximation*
- *mioFeaspumpLevel*
- *mioMaxNumRootCutRounds*
- *mioPropagateObjectiveConstraint*
- *mioSeed*
- *presolveMaxNumPass*
- *simSeed*

Removed

- *dataTolAij*
- *intpntNlMeritBal*
- *intpntNlTolDfeas*
- *intpntNlTolMuRed*
- *intpntNlTolNearRel*
- *intpntNlTolPfeas*
- *intpntNlTolRelGap*
- *intpntNlTolRelStep*
- *mioDisableTermTime*
- *mioNearTolAbsGap*
- *mioNearTolRelGap*
- *mioConstructSol*
- *mioMtUserCb*
- *opfMaxTermsPerLine*
- *readDataCompressed*
- *readDataFormat*
- *writeDataCompressed*
- *writeDataFormat*

17.3 Constants

Added

Removed

- xml
- mioHeuristicTime
- mioOptimizerTime
- mioConstructNumRoundings
- mioInitialSolution
- mioNearAbsgapSatisfied
- mioNearRelgapSatisfied
- mioSimMaxiterSetbacks
- hybrid
- worst
- geco
- nearDualFeas
- nearPrimAndDualFeas
- nearPrimFeas
- optimalPartition
- nearDualFeas
- nearDualInfeasCer
- nearIntegerOptimal
- nearOptimal
- nearPrimAndDualFeas
- nearPrimFeas
- nearPrimInfeasCer

Bibliography

- [AA95] E. D. Andersen and K. D. Andersen. Presolving in linear programming. *Math. Programming*, 71(2):221–245, 1995.
- [AGMX96] E. D. Andersen, J. Gondzio, Cs. Mészáros, and X. Xu. Implementation of interior point methods for large scale linear programming. In T. Terlaky, editor, *Interior-point methods of mathematical programming*, pages 189–252. Kluwer Academic Publishers, 1996.
- [ART03] E. D. Andersen, C. Roos, and T. Terlaky. On implementing a primal-dual interior-point method for conic quadratic optimization. *Math. Programming*, February 2003.
- [AY96] E. D. Andersen and Y. Ye. Combining interior-point and pivoting algorithms. *Management Sci.*, 42(12):1719–1731, December 1996.
- [And09] Erling D. Andersen. The homogeneous and self-dual model and algorithm for linear optimization. Technical Report TR-1-2009, MOSEK ApS, 2009. URL: <http://docs.mosek.com/whitepapers/homolo.pdf>.
- [And13] Erling D. Andersen. On formulating quadratic functions in optimization models. Technical Report TR-1-2013, MOSEK ApS, 2013. Last revised 23-feb-2016. URL: <http://docs.mosek.com/whitepapers/qmodel.pdf>.
- [BKVH07] S. Boyd, S.J. Kim, L. Vandenberghe, and A. Hassibi. A Tutorial on Geometric Programming. *Optimization and Engineering*, 8(1):67–127, 2007. Available at http://www.stanford.edu/~boyd/gp_tutorial.html.
- [Chv83] V. Chvátal. *Linear programming*. W.H. Freeman and Company, 1983.
- [GJ79] Michael R Gary and David S Johnson. Computers and intractability: a guide to the theory of np-completeness. 1979.
- [GY05] Donald Goldfarb and Wotao Yin. Second-order cone programming methods for total variation-based image restoration. *SIAM Journal on Scientific Computing*, 27(2):622–645, 2005.
- [Gra69] Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM journal on Applied Mathematics*, 17(2):416–429, 1969.
- [GK00] Richard C. Grinold and Ronald N. Kahn. *Active portfolio management*. McGraw-Hill, New York, 2 edition, 2000.
- [Naz87] J. L. Nazareth. *Computer Solution of Linear Programs*. Oxford University Press, New York, 1987.
- [PB15] Jaehyun Park and Stephen Boyd. A semidefinite programming method for integer convex quadratic minimization. *arXiv preprint arXiv:1504.07672*, 2015.
- [Pat03] Gábor Pataki. Teaching integer programming formulations using the traveling salesman problem. *SIAM review*, 45(1):116–123, 2003.
- [Wol98] L. A. Wolsey. *Integer programming*. John Wiley and Sons, 1998.
- [BenTalN01] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. MPS/SIAM Series on Optimization. SIAM, 2001.

Symbol Index

Classes

`ndarray<T,N>`, 146
`rc_ptr<T>`, 149
`shape_t<N>`, 148
`BaseExpression`, 152
`BaseVariable`, 154
`ConeDomain`, 158
`ConicConstraint`, 159
`ConicVariable`, 159
`Constraint`, 160
`Domain`, 163
`Expr`, 171
`Expression`, 180
`FlatExpr`, 182
`LinearConstraint`, 183
`LinearDomain`, 183
`LinearPSDConstraint`, 184
`LinearPSDVariable`, 184
`LinearVariable`, 185
`LinPSDDomain`, 183
`Matrix`, 186
`Model`, 190
`ModelConstraint`, 199
`ModelVariable`, 199
`NDSparseArray`, 200
`PSDConstraint`, 200
`PSDDomain`, 201
`PSDVariable`, 201
`RangedConstraint`, 203
`RangeDomain`, 202
`RangedVariable`, 203
`Set`, 204
`SliceConstraint`, 205
`SliceVariable`, 205
`SymLinearVariable`, 206
`SymmetricExpr`, 207
`SymmetricLinearDomain`, 208
`SymmetricRangeDomain`, 208
`SymmetricVariable`, 208
`SymRangedVariable`, 207
`Var`, 209
`Variable`, 213
`WorkStack`, 216
`LinAlg`, 280

Enumerations

`AccSolutionStatus`, 253
`AccSolutionStatus.Optimal`, 253

`AccSolutionStatus.Feasible`, 253
`AccSolutionStatus.Certificate`, 253
`AccSolutionStatus.Anything`, 253
`ObjectiveSense`, 253
`ObjectiveSense.Undefined`, 253
`ObjectiveSense.Minimize`, 253
`ObjectiveSense.Maximize`, 253
`ProblemStatus`, 253
`ProblemStatus.Unknown`, 253
`ProblemStatus.PrimalInfeasibleOrUnbounded`, 253
`ProblemStatus.PrimalInfeasible`, 253
`ProblemStatus.PrimalFeasible`, 253
`ProblemStatus.PrimalAndDualInfeasible`, 253
`ProblemStatus.PrimalAndDualFeasible`, 253
`ProblemStatus.IllPosed`, 253
`ProblemStatus.DualInfeasible`, 253
`ProblemStatus.DualFeasible`, 253
`SolutionStatus`, 253
`SolutionStatus.Unknown`, 253
`SolutionStatus.Undefined`, 253
`SolutionStatus.Optimal`, 253
`SolutionStatus.IllposedCert`, 254
`SolutionStatus.Feasible`, 253
`SolutionStatus.Certificate`, 254
`SolutionType`, 254
`SolutionType.Interior`, 254
`SolutionType.Integer`, 254
`SolutionType.Default`, 254
`SolutionType.Basic`, 254

Exceptions

`monty::ArrayInitializerException`, 149
`DimensionError`, 277
`DomainError`, 277
`ExpressionError`, 277
`FatalError`, 277
`FusionException`, 277
`FusionRuntimeException`, 277
`IndexError`, 278
`IOError`, 278
`LengthError`, 278
`MatrixError`, 278
`ModelError`, 278
`NameError`, 278
`OptimizeError`, 279
`ParameterError`, 279
`RangeError`, 279
`SetDefinitionError`, 279

- SliceError, 279
- SolutionError, 279
- SparseFormatError, 279
- UnexpectedError, 279
- UnimplementedError, 280
- ValueConversionError, 280

Parameters

- Double parameters, 227
- basisRelTolS, 227
- basisTolS, 227
- basisTolX, 227
- intpntCoTolDfeas, 227
- intpntCoTolInfeas, 227
- intpntCoTolMuRed, 227
- intpntCoTolNearRel, 228
- intpntCoTolPfeas, 228
- intpntCoTolRelGap, 228
- intpntTolDfeas, 228
- intpntTolDsafe, 228
- intpntTolInfeas, 229
- intpntTolMuRed, 229
- intpntTolPath, 229
- intpntTolPfeas, 229
- intpntTolPsafe, 229
- intpntTolRelGap, 229
- intpntTolRelStep, 230
- intpntTolStepSize, 230
- lowerObjCut, 230
- lowerObjCutFiniteTrh, 230
- mioMaxTime, 230
- mioRelGapConst, 230
- mioTolAbsGap, 231
- mioTolAbsRelaxInt, 231
- mioTolFeas, 231
- mioTolRelDualBoundImprovement, 231
- mioTolRelGap, 231
- optimizerMaxTime, 231
- presolveTolAbsLindep, 232
- presolveTolAij, 232
- presolveTolRelLindep, 232
- presolveTolS, 232
- presolveTolX, 232
- simLuTolRelPiv, 232
- simplexAbsTolPiv, 232
- upperObjCut, 233
- upperObjCutFiniteTrh, 233
- Integer parameters, 233
- autoUpdateSolInfo, 233
- biCleanOptimizer, 233
- biIgnoreMaxIter, 233
- biIgnoreNumError, 234
- biMaxIterations, 234
- cacheLicense, 234
- infeasPreferPrimal, 234
- intpntBasis, 234
- intpntDiffStep, 235
- intpntMaxIterations, 235
- intpntMaxNumCor, 235
- intpntMultiThread, 235
- intpntOffColTrh, 235
- intpntOrderGpNumSeeds, 235
- intpntOrderMethod, 236
- intpntRegularizationUse, 236
- intpntScaling, 236
- intpntSolveForm, 236
- intpntStartingPoint, 236
- licenseDebug, 236
- licensePauseTime, 237
- licenseSuppressExpireWrns, 237
- licenseTrhExpiryWrn, 237
- licenseWait, 237
- log, 237
- logBi, 237
- logBiFreq, 238
- logCutSecondOpt, 238
- logExpand, 238
- logFile, 238
- logInfeasAna, 238
- logIntpnt, 239
- logLocalInfo, 239
- logMio, 239
- logMioFreq, 239
- logOrder, 239
- logPresolve, 239
- logResponse, 240
- logSim, 240
- logSimFreq, 240
- logSimMinor, 240
- mioBranchDir, 240
- mioConicOuterApproximation, 240
- mioCutClique, 241
- mioCutCmir, 241
- mioCutGmi, 241
- mioCutImpliedBound, 241
- mioCutKnapsackCover, 241
- mioCutSelectionLevel, 241
- mioFeaspumpLevel, 242
- mioHeuristicLevel, 242
- mioMaxNumBranches, 242
- mioMaxNumRelaxs, 242
- mioMaxNumRootCutRounds, 242
- mioMaxNumSolutions, 243
- mioMode, 243
- mioNodeOptimizer, 243
- mioNodeSelection, 243
- mioPerspectiveReformulate, 243
- mioProbingLevel, 243
- mioPropagateObjectiveConstraint, 244
- mioRinsMaxNodes, 244
- mioRootOptimizer, 244
- mioRootRepeatPresolveLevel, 244
- mioSeed, 244
- mioVbDetectionLevel, 245
- mtSpincount, 245
- numThreads, 245

- optimizer, 245
- presolveEliminatorMaxFill, 245
- presolveEliminatorMaxNumTries, 246
- presolveLevel, 246
- presolveLindepAbsWorkTrh, 246
- presolveLindepRelWorkTrh, 246
- presolveLindepUse, 246
- presolveMaxNumPass, 246
- presolveUse, 247
- removeUnusedSolutions, 247
- simBasisFactorUse, 247
- simDegen, 247
- simDualCrash, 247
- simDualPhaseoneMethod, 247
- simDualRestrictSelection, 248
- simDualSelection, 248
- simExploitDupvec, 248
- simHotstart, 248
- simHotstartLu, 248
- simMaxIterations, 248
- simMaxNumSetbacks, 249
- simNonSingular, 249
- simPrimalCrash, 249
- simPrimalPhaseoneMethod, 249
- simPrimalRestrictSelection, 249
- simPrimalSelection, 249
- simRefactorFreq, 250
- simReformulation, 250
- simSaveLu, 250
- simScaling, 250
- simScalingMethod, 250
- simSeed, 250
- simSolveForm, 251
- simSwitchOptimizer, 251
- writeLpFullObj, 251
- writeLpLineWidth, 251
- writeLpQuotedNames, 251
- writeLpTermsPerLine, 251
- String parameters, 252
- basSolFileName, 252
- dataFileName, 252
- intSolFileName, 252
- itrSolFileName, 252
- remoteAccessToken, 252
- writeLpGenVarName, 252

Response codes

Index

A

- algorithm
 - approximation, 94, 117
- approximation
 - algorithm, 94, 117
 - correlation matrix, 112
- asset, *see* portfolio optimization
- assignment problem, 103

B

- basic
 - solution, 44
- basis identification, 133
- bound
 - constraint, 18, 119, 122
 - linear optimization, 18
 - variable, 18, 119, 122

C

- callback, 54
- cardinality constraints, 39, 84
- CBF format, 309
- certificate, 45
 - dual, 121, 125
 - primal, 120, 124
- Cholesky factorization, 80
- compile
 - Linux, examples, 7
- complementarity, 120, 124
- cone, 11
 - dual, 123
 - dual exponential, 25
 - exponential, 25
 - power, 23, 101
 - quadratic, 20
 - rotated quadratic, 20
 - semidefinite, 27
- conic exponential optimization, 25
- conic optimization, 20, 23, 25, 122
 - interior-point, 136
 - modeling, 11
 - termination criteria, 138
- conic quadratic optimization, 20
- constraint
 - bound, 18, 119, 122
 - linear optimization, 18
 - matrix, 18, 119, 122
 - modeling, 13
- correlation matrix, 75, 112

- approximation, 112

- covariance matrix, *see* correlation matrix
- cut, 141

D

- denoising, 91
- dense
 - matrix, 72
- determinant, 101
- determinism, 71
- dual
 - certificate, 121, 125
 - cone, 123
 - feasible, 120
 - infeasible, 120, 121, 125
 - problem, 120, 123, 126
 - solution, 22, 24, 27, 46
 - variable, 120, 123
- duality
 - conic, 123
 - linear, 120
 - semidefinite, 126
- dualizer, 129

E

- efficient frontier, 78
- eliminator, 129
- ellipsoid, 98
- entropy, 37
 - relative, 37
- error
 - optimization, 44
- errors, 48
- examples
 - compile Linux, 7
- exceptions, 48
- exponential, 37
- exponential cone, 25
- expression
 - modeling, 12

F

- factor model, 80, 112
- feasibility problem, 103
- feasible
 - dual, 120
 - primal, 119, 131, 137
 - problem, 119
- format, 50

- CBF, 309
- json, 328
- LP, 284
- MPS, 289
- OPF, 300
- PTF, 323
- sol, 335
- task, 328
- Frobenius norm, 112
- Fusion
 - reformulation, 10
- G
 - geometric mean, 36, 101
 - geometric programming, 33
 - GP, 33
- H
 - hot-start, 135
- I
 - I/O, 50
 - infeasibility, 45, 120, 124
 - linear optimization, 120
 - semidefinite, 126
 - infeasible
 - dual, 120, 121, 125
 - primal, 119, 120, 124, 131, 138
 - problem, 119, 120, 126
 - information item, 52, 54
 - installation, 5
 - makefile, 7
 - requirements, 5
 - troubleshooting, 5
 - Visual Studio, 8
 - integer
 - optimizer, 140
 - solution, 44
 - variable, 29
 - integer feasible
 - solution, 142
 - integer optimization, 29, 140
 - cut, 141
 - initial solution, 31, 94
 - objective bound, 141
 - optimality gap, 143
 - parameter, 30
 - relaxation, 141
 - termination criteria, 142
 - tolerance, 142
 - integer optimizer
 - logging, 143
 - interior-point
 - conic optimization, 136
 - linear optimization, 130
 - logging, 134, 140
 - optimizer, 130, 136
 - solution, 44

- termination criteria, 132, 138
- J
 - json format, 328
- L
 - Löwner-John ellipsoid, 98
 - least squares
 - integer, 115
 - license, 73
 - checkout, 73
 - parameter, 73
 - path, 73
 - limitations, 70
 - linear constraint matrix, 18
 - linear dependency, 129
 - linear optimization, 18, 119
 - bound, 18
 - constraint, 18
 - infeasibility, 120
 - interior-point, 130
 - objective, 18
 - simplex, 135
 - termination criteria, 132, 135
 - variable, 18
 - Linux
 - examples compile, 7
 - log-sum-exp, 37, 97
 - logarithm, 37
 - logging, 50
 - integer optimizer, 143
 - interior-point, 134, 140
 - optimizer, 134, 136, 140
 - simplex, 136
 - logistic regression, 97
 - LP format, 284
- M
 - machine learning
 - large margin classification, 86
 - logistic regression, 97
 - separating hyperplane, 86
 - Support-Vector Machine, 86
 - makespan, 93
 - market impact cost, 81
 - Markowitz model, 75
 - matrix
 - constraint, 18, 119, 122
 - dense, 72
 - low rank, 114
 - modeling, 14
 - semidefinite, 27
 - sparse, 72
 - symmetric, 27
 - memory management, 70
 - MIP, *see* integer optimization
 - mixed-integer, *see* integer

- mixed-integer optimization, *see* integer optimization
- modeling
 - conic optimization, 11
 - constraint, 13
 - design, 8
 - expression, 12
 - matrix, 14
 - objective, 13
 - variable, 11
- monomial, 36
- MPS format, 289
 - free, 300
- N**
- near-optimal
 - solution, 142
- norm
 - 1-norm, 35
 - 2-norm, 36
 - Frobenius, 112
 - nuclear, 114
 - p-norm, 36
- nuclear norm, 114
- numerical issues
 - presolve, 129
 - scaling, 129
 - simplex, 135
- O**
- objective, 119, 122
 - linear optimization, 18
 - modeling, 13
- objective bound, 141
- OPF format, 300
- optimal
 - solution, 45
- optimality gap, 143
- optimization
 - conic, 122
 - conic quadratic, 122
 - error, 44
 - linear, 18, 119
 - semidefinite, 126
- optimizer
 - determinism, 71
 - integer, 140
 - interior-point, 130, 136
 - interrupt, 53, 54
 - logging, 134, 136, 140
 - selection, 129, 130
 - simplex, 135
 - time limit, 53
- Optimizer API, 56
 - reformulation, 10
- P**
- parallelization, 71
- parameter, 51
 - integer optimization, 30
 - license, 73
 - simplex, 135
- Pareto optimality, 75
- path
 - license, 73
- penalty, 86
- portfolio optimization, 75
 - cardinality constraints, 39, 84
 - correlation matrix, 112
 - efficient frontier, 78
 - factor model, 80, 112
 - market impact cost, 81
 - Markowitz model, 75
 - Pareto optimality, 75
 - slippage cost, 81
 - transaction cost, 83
- power, 36
- power cone, 23, 101
- power cone optimization, 23
- presolve, 128
 - eliminator, 129
 - linear dependency check, 129
 - numerical issues, 129
- primal
 - certificate, 120, 124
 - feasible, 119, 131, 137
 - infeasible, 119, 120, 124, 131, 138
 - problem, 120, 123, 126
 - solution, 22, 24, 27, 46, 119
- primal-dual
 - problem, 130, 137
 - solution, 120
- problem
 - dual, 120, 123, 126
 - feasible, 119
 - infeasible, 119, 120, 126
 - load, 51
 - primal, 120, 123, 126
 - primal-dual, 130, 137
 - save, 50
 - status, 44
 - unbounded, 121, 125
- PTF format, 323
- Q**
- quadratic cone, 20
- quality
 - solution, 143
- R**
- regression
 - logistic, 97
- relaxation, 115, 141
- reoptimization, 16, 108
- response code, 48
- rotated quadratic cone, 20

S

- scaling, 129
- scheduling, 93
- Schur complement, 116
- semicontinuous variable, 38
- semidefinite
 - cone, 27
 - infeasibility, 126
 - matrix, 27
 - variable, 27
- semidefinite optimization, 27, 126
- separating hyperplane, 86
- simplex
 - linear optimization, 135
 - logging, 136
 - numerical issues, 135
 - optimizer, 135
 - parameter, 135
 - termination criteria, 135
- slice
 - variable, 15, 72, 91
- slippage cost, 81
- softplus, 37
- sol format, 335
- solution
 - basic, 44
 - dual, 22, 24, 27, 46
 - file format, 335
 - integer, 44
 - integer feasible, 142
 - interior-point, 44
 - near-optimal, 142
 - optimal, 45
 - primal, 22, 24, 27, 46, 119
 - primal-dual, 120
 - quality, 143
 - retrieve, 44
 - status, 45
- sparse
 - matrix, 72
- stacking, 15
- status
 - problem, 44
 - solution, 45
- symmetric
 - matrix, 27

T

- task format, 328
- termination, 44
- termination criteria, 54
 - conic optimization, 138
 - integer optimization, 142
 - interior-point, 132, 138
 - linear optimization, 132, 135
 - simplex, 135
 - tolerance, 133, 139, 142
- thread, 71

- time limit, 53, 54
- tolerance
 - integer optimization, 142
 - termination criteria, 133, 139, 142
- transaction cost, 83
- travelling salesman problem, 107
- troubleshooting
 - installation, 5

U

- unbounded
 - problem, 121, 125
- user callback, *see* callback

V

- variable, 119, 122
 - bound, 18, 119, 122
 - dual, 120, 123
 - integer, 29
 - limitations, 70
 - linear optimization, 18
 - modeling, 11
 - semicontinuous, 38
 - semidefinite, 27
 - slice, 15, 72, 91
- vectorization, 16, 72
- Visual Studio
 - installation, 8