16.2 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]*.

16.2.1 MPS File Structure

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

$$\begin{array}{rcccl} \text{maximize/minimize} & & c^T x + q_0(x) & & \\ l^c & \leq & Ax + q(x) & \leq & u^c, \\ l^x & \leq & x & \leq & u^x, \\ & & x \in \mathcal{K}, & & \\ & & x_{\mathcal{J}} \text{ integer}, & & \end{array} \tag{16.2}$$

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.

$q : \mathbb{R}^n \to \mathbb{R}$ 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.

$\mathcal{J} \subseteq \{1, 2, \ldots, n\}$ 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
*234567890123456789012345678901234567890123456789012345678901234567890
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]
```

(continues on next page)

(continued from previous page)

```
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

```
[+|-]XXXXXXX.XXXXXX[[e|E][+|-]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. 16.2.5 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
```

(continues on next page)

(continued from previous page)

```
    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 -h 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]
```

(continued from previous page)

```
    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 QSECTIONs is allowed for each constraint.
- The QSECTIONs can appear in an arbitrary order after the COLUMNS section.
- All variable names occurring in the QSECTIONs must already be speci ed in the COLUMNS section.
- All entries speci ed in a QSECTIONs 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 de ne the quadratic term of the objective function. They di er in how the quadratic term of the objective function is stored:

- QMATRIX stores all the nonzeros coe cients, 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 eld 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 speci es 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 o -diagonal coe cient 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

$$\begin{array}{lll} \text{minimize} & x_2 + \frac{1}{2}(2x_1^2 \quad 2x_1x_3 + 0:2x_2^2 + 2x_3^2) & \\ \text{subject to} & x_1 + x_2 + x_3 & 1; \\ & x \quad 0 & \end{array}$$

has the following MPS le representation using QMATRIX

```
* File:  qo1_matrix.mps
NAME            qo1_qmatrix
ROWS
 N   obj
 G   c1
COLUMNS
    x1          c1          1.0
    x2          obj         -1.0
```

(continues on next page)

(continued from previous page)

```
    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
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.

QCMATRIX (optional)

A QCMATRIX section allows to specify the quadratic part of a given constraint. Within the QCMATRIX 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^i_{kj} is assigned the value given by [value1]. Moreover, the quadratic term is represented as $1/2x^TQx$.

The example

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

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
QCMATRIX      q1
    x1        x1        2.0
    x1        x3        -1.0
    x3        x1        -1.0
    x2        x2        0.2
    x3        x3        2.0
ENDATA
```

Regarding the `QCMATRIX`s please note that:

Only one `QCMATRIX` is allowed for each constraint.

The `QCMATRIX`s 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 (16.2). It is assumed that $\mathcal{K}$ satisfies the following requirements. Let

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

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

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

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

$\mathbb{R}$ set:

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

Zero cone:

$$\mathcal{K}_t = \{0\} \subseteq \mathbb{R}^{n^t}. \tag{16.3}$$

Quadratic cone:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\}. \tag{16.4}$$

Rotated quadratic cone:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : 2x_1x_2 \geq \sum_{j=3}^{n^t} x_j^2, \quad x_1, x_2 \geq 0 \right\}. \tag{16.5}$$

Primal exponential cone:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^3 : x_1 \geq x_2 \exp(x_3/x_2), \quad x_1, x_2 \geq 0 \right\}. \tag{16.6}$$

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

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : x_1^\alpha x_2^{1-\alpha} \geq \sqrt{\sum_{j=3}^{n^t} x_j^2}, \quad x_1, x_2 \geq 0 \right\}. \tag{16.7}$$

Dual exponential cone:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^3 : x_1 \geq -x_3 e^{-1} \exp(x_2/x_3), \quad x_3 \leq 0, x_1 \geq 0 \right\}. \tag{16.8}$$

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

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : \left(\frac{x_1}{\alpha}\right)^{\alpha} \left(\frac{x_2}{1-\alpha}\right)^{1-\alpha} \geq \sqrt{\sum_{j=3}^{n^t} x_j^2}, \quad x_1, x_2 \geq 0 \right\}. \tag{16.9}$$

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
*234567890123456789012345678901234567890123456789012345678901234567890
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 (16.3).
`QUAD`	≥ 1	unused	Quadratic cone (16.4).
`RQUAD`	≥ 2	unused	Rotated quadratic cone (16.5).
`PEXP`	3	unused	Primal exponential cone (16.6).
`PPOW`	≥ 2	α	Primal power cone (16.7).
`DEXP`	3	unused	Dual exponential cone (16.8).
`DPOW`	≥ 2	α	Dual power cone (16.9).

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.

16.2.2 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.

16.2.3 General Limitations

An MPS file should be an ASCII file.

16.2.4 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.

16.2.5 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.

Moreover, by default a line in the MPS file must not contain more than 1024 characters. By modifying the parameter *MSK_IPAR_READ_MPS_WIDTH* an arbitrary large line width will be accepted.

The free MPS format is default. To change to the strict and other formats use the parameter *MSK_IPAR_READ_MPS_FORMAT*.

16.3 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.

16.3.1 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]
```

16.3.2 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 $x_1, \ldots, x_n$ 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 $x_1, \ldots, x_n$ 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 $x_1, \ldots, x_n$ 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 $x_1, \ldots, x_n$ defines a constraint of the form

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

`zero`: zero cone of n variables $x_1, \ldots, x_n$ defines a constraint of the form

$$x_1 = \cdots = x_n = 0$$

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.

sl 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, sl and su. Items sl and su are not required for integer solutions.

A [con] section should always contain sk, lvl, sl, 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  sl=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.

16.3.3 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]*|[.][0-9]+)([eE][+|-]?[0-9]+)?
```

16.3.4 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"
```

16.3.5 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]
```

16.3.6 Writing OPF Files from MOSEK

To write an OPF file then make sure the file extension is .opf.

Then modify the following parameters to define what the file should contain:

MSK_IPAR_OPF_WRITE_SOL_BAS	Include basic solution, if defined.
MSK_IPAR_OPF_WRITE_SOL_ITG	Include integer solution, if defined.
MSK_IPAR_OPF_WRITE_SOL_ITR	Include interior solution, if defined.
MSK_IPAR_OPF_WRITE_SOLUTIONS	Include solutions if they are defined. If this is off, no solutions are included.
MSK_IPAR_OPF_WRITE_HEADER	Include a small header with comments.
MSK_IPAR_OPF_WRITE_PROBLEM	Include the problem itself — objective, constraints and bounds.
MSK_IPAR_OPF_WRITE_PARAMETERS	Include all parameter settings.
MSK_IPAR_OPF_WRITE_HINTS	Include hints about the size of the problem.

16.3.7 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:

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

having the bounds

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

In the `OPF` format the example is displayed as shown in Listing 16.1.

Listing 16.1: 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]
```

(continues on next page)

(continued from previous page)

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

Quadratic Example `qo1.opf`

An example of a quadratic optimization problem is

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

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

Listing 16.2: 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.

   - 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:

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

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 16.3.

Listing 16.3: 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
  [cone rquad 'k2'] x5, x6, x3 [/cone]
[/bounds]
```

Mixed Integer Example `milo1.opf`

Consider the mixed integer problem:

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

This can be implemented in `OPF` with the file in Listing 16.4.

Listing 16.4: Example of an OPF file for a mixed-integer linear problem.

```
[comment]
  The milo1 example in OPF format
[/comment]

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

(continues on next page)

(continued from previous page)

```
[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]
```

16.4 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.

16.4.1 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:

$$\begin{array}{lll} \min / \max & g^{obj} & \\ & g_i \in \mathcal{K}_i, & i \in \mathcal{I}, \\ \text{s.t.} & G_i \in \mathcal{K}_i, & i \in \mathcal{I}^{PSD}, \\ & x_j \in \mathcal{K}_j, & j \in \mathcal{J}, \\ & \overline{X}_j \in \mathcal{K}_j, & j \in \mathcal{J}^{PSD}. \end{array} \tag{16.10}$$

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 $i \in \mathcal{I}^{PSD}$

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

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

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

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

Negative orthant - A cone in the linear family defined by

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

Fixpoint zero - A cone in the linear family defined by

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

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

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

16.4.2 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 seperator 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.

16.4.3 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, $\{0, 1, \ldots\}$, 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

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

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 16.3. 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

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

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 16.3.

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 16.3: 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

16.4.4 File Format Keywords

VER

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

`HEADER`: None

`BODY`: One line formatted as:

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

16.4.5 CBF Format Examples

Minimal Working Example

The conic optimization problem (16.11) , has three variables in a quadratic cone - first one is integer - and an affine expression in domain 0 (equality constraint).

$$\begin{array}{ll} \text{minimize} & 5.1\,x_0 \\ \text{subject to} & 6.2\,x_1 + 7.3\,x_2 - 8.4 \in \{0\} \\ & x \in \mathcal{Q}^3,\ x_0 \in \mathbb{Z}. \end{array} \tag{16.11}$$

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
```

(continues on next page)

(continued from previous page)

```
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.

```
OBJACOORD
1
0 5.1

ACOORD
2
0 1 6.2
0 2 7.3

BCOORD
1
0 -8.4
```

This concludes the example.

Mixing Linear, Second-order and Semidefinite Cones

The conic optimization problem (16.12), has a semidefinite cone, a quadratic cone over unordered subindices, and two equality constraints.

$$\begin{array}{lll} \text{minimize} & \left\langle \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}, X_1 \right\rangle + x_1 & \\ \text{subject to} & \left\langle \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, X_1 \right\rangle + x_1 & = \quad 1.0\,, \\ & \left\langle \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, X_1 \right\rangle + x_0 + x_2 & = \quad 0.5\,, \\ & x_1 \geq \sqrt{x_0^2 + x_2^2}\,, & \\ & X_1 \succeq \mathbf{0}\,. & \end{array} \tag{16.12}$$

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
```

(continues on next page)

(continued from previous page)

```
# 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
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...
ACOORD
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
```

(continues on next page)

(continued from previous page)

```
#     | F^{obj}[0][0,0] = 1.0
#     | F^{obj}[0][1,1] = 1.0
OBJFCOORD
2
0 0 0 1.0
0 1 1 1.0

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

# One coordinate in b^{obj} coefficient:
#     | b^{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 (16.14), is defined for a sequence of objectives such that hotstarting from one to the next might be advantages.

$$\begin{array}{lll} \text{maximize}_k & g_k^{obj} & \\ \text{subject to} & 50\,x_0 + 31 & \leq \quad 250\,, \\ & 3\,x_0 - 2x_1 & \geq \quad -4\,, \\ & x \in \mathbb{R}^2_+, & \end{array} \tag{16.14}$$

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 16.5.

Listing 16.5: Problem (16.14) 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
OBJACOORD
2
0 1.0
1 0.64

# Four coordinates in a_ij coefficients:
#     | a[0,0] = 50.0
#     | a[1,0] = 3.0
#     | and more...
ACOORD
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
BCOORD
2
0 -250.0
1 4.0
```

(continues on next page)

(continued from previous page)

```
# 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
OBJACOORD
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
OBJACOORD
1
1 0.85
```

16.5 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.

16.5.1 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 that 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
```

16.5.2 `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.

16.5.3 `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 >
```

16.5.4 `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 separete 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. It 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*(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
```

16.5.5 `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
```

16.5.6 `Integer` section

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

```
Integer
   x1 x2 x3
```

16.5.7 `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)
```

16.5.8 `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" WHICHSOL
      "ProblemStatus" PROSTA PROSTA?
   "SolutionStatus" SOLSTA SOLSTA?
   "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:

16.7 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.

16.7.1 *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**.

16.7.2 *jsol* format

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

16.7.3 A *jtask* example

In Listing 16.6 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 16.6: 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",
```

(continues on next page)

(continued from previous page)


```
                    1e+0,
                    5e+0,
                    1e+0
                ]
            },
            "cfix":0.0
        },
        "A":{
            "subi":[
                0,
                0,
                0,
                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,
```


(continues on next page)

(continued from previous page)


```
            "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,
```


(continues on next page)

(continued from previous page)


```
"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_REDUCTIONS":-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,
```


(continues on next page)

(continued from previous page)

```
            "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,
```

(continues on next page)

(continued from previous page)

```
            "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":"",
```

(continues on next page)

(continued from previous page)

```
            "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"
        }
    }
}
```

16.8 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

$$l_i^c \leq \sum_{j=1}^{n} a_{ij} x_j \leq u_i^c, \tag{16.15}$$

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 16.4 the possible values of the status keys and their interpretation are shown.

Table 16.4: 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 (16.15).)
- `UPPER LIMIT`: the quantity u_i^c (see (16.15).)
- `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 16.7 we show the solution file for the `lo1.opf` problem.

Listing 16.7: 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
```

(continues on next page)

(continued from previous page)

```
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 17

List of examples

List of examples shipped in the distribution of Optimization Toolbox for MATLAB:

Table 17.1: List of distributed examples

File	Description
`advs.m`	Advanced simplex hot-start examples
`affco1.m`	A simple problem using affine conic constraints
`affco2.m`	A simple problem using affine conic constraints
`callback.m`	An example of data/progress callback
`callback_handler.m`	Log handler definition for `callback.m`
`ceo1.m`	A simple conic exponential problem
`cqo1.m`	A simple conic quadratic problem
`feasrepairex1.m`	A simple example of how to repair an infeasible problem
`gp1.m`	A simple geometric program (GP) in conic form
`lo1.m`	A simple linear problem using msklpopt
`lo2.m`	A simple linear problem using mosekopt
`lo3.m`	A simple linear problem using linprog
`mico1.m`	A simple mixed-integer conic problem
`milo1.m`	A simple mixed-integer linear problem
`mioinitsol.m`	A simple mixed-integer linear problem with an initial guess
`normex.m`	Demonstrates least squares and other norm minimization problems
`parameters.m`	Shows how to set optimizer parameters and read information items
`portfolio_1_basic.m`	Portfolio optimization - basic Markowitz model
`portfolio_2_frontier.m`	Portfolio optimization - efficient frontier
`portfolio_3_impact.m`	Portfolio optimization - market impact costs
`portfolio_4_transcost.m`	Portfolio optimization - transaction costs
`portfolio_5_card.m`	Portfolio optimization - cardinality constraints
`pow1.m`	A simple power cone problem
`qcqo1.m`	A simple quadratically constrained quadratic problem
`qo1.m`	A simple quadratic problem
`qo2.m`	A simple quadratic problem
`reoptimization.m`	Demonstrate how to modify and re-optimize a linear problem
`response.m`	Demonstrates proper response handling
`rlo1.m`	Robust linear optimization example, part 1

Continued on next page

Table 17.1 – continued from previous page

File	Description
`rlo2.m`	Robust linear optimization example, part 2
`sdo1.m`	A simple semidefinite optimization problem
`sensitivity.m`	Sensitivity analysis performed on a small linear problem
`sensitivity2.m`	Sensitivity analysis performed on a small linear problem
`simple.m`	A simple I/O example: read problem from a file, solve and write solutions
`solutionquality.m`	Demonstrates how to examine the quality of a solution

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

Chapter 18

Interface changes

The section shows interface-specific changes to the **MOSEK** Optimization Toolbox for MATLAB in version 9.0. See the release notes for general changes and new features of the **MOSEK** Optimization Suite.

18.1 Backwards compatibility

Remove all `Near` solution statuses i.e. `MSK_SOL_STA_NEAR_OPTIMAL`, `MSK_SOL_STA_NEAR_PRIM_INFEAS_CER`, etc. See Sec. 13.3.3.

All functions related to the general nonlinear optimizer and Scopt have been removed. See Sec. 15.8.

18.2 New API

Introduced a possibility to specify *affine conic constraints* i.e. conic constraints of the form $Fx + g \in K$ directly. See Sec. 6.7 and Sec. 12.5 for details.

18.3 Parameters

Added

MSK_IPAR_INTPNT_PURIFY

MSK_IPAR_MIO_CONIC_OUTER_APPROXIMATION

MSK_IPAR_MIO_PROPAGATE_OBJECTIVE_CONSTRAINT

MSK_IPAR_MIO_SEED

MSK_IPAR_OPF_WRITE_LINE_LENGTH

MSK_IPAR_PRESOLVE_MAX_NUM_PASS

MSK_IPAR_SIM_SEED

MSK_IPAR_WRITE_COMPRESSION

Removed

`MSK_DPAR_DATA_TOL_AIJ`

`MSK_DPAR_INTPNT_NL_MERIT_BAL`

`MSK_DPAR_INTPNT_NL_TOL_DFEAS`

MSK_DPAR_INTPNT_NL_TOL_MU_RED

MSK_DPAR_INTPNT_NL_TOL_NEAR_REL

MSK_DPAR_INTPNT_NL_TOL_PFEAS

MSK_DPAR_INTPNT_NL_TOL_REL_GAP

MSK_DPAR_INTPNT_NL_TOL_REL_STEP

MSK_DPAR_MIO_DISABLE_TERM_TIME

MSK_DPAR_MIO_NEAR_TOL_ABS_GAP

MSK_DPAR_MIO_NEAR_TOL_REL_GAP

MSK_IPAR_MIO_CONSTRUCT_SOL

MSK_IPAR_MIO_MT_USER_CB

MSK_IPAR_OPF_MAX_TERMS_PER_LINE

MSK_IPAR_READ_DATA_COMPRESSED

MSK_IPAR_READ_DATA_FORMAT

MSK_IPAR_WRITE_DATA_COMPRESSED

MSK_IPAR_WRITE_DATA_FORMAT

18.4 Constants

Added

"MSK_COMPRESS_ZSTD"

"MSK_CT_DEXP"

"MSK_CT_DPOW"

"MSK_CT_PEXP"

"MSK_CT_PPOW"

"MSK_CT_ZERO"

"MSK_DATA_FORMAT_PTF"

"MSK_IINF_MIO_NUMBIN"

"MSK_IINF_MIO_NUMBINCONEVAR"

"MSK_IINF_MIO_NUMCONE"

"MSK_IINF_MIO_NUMCONEVAR"

"MSK_IINF_MIO_NUMCONT"

"MSK_IINF_MIO_NUMCONTCONEVAR"

"MSK_IINF_MIO_NUMINTCONEVAR"

"MSK_IINF_MIO_NUMPEXPCONES"

"MSK_IINF_MIO_NUMQCONES"

"MSK_IINF_MIO_NUMRQCONES"

"MSK_IINF_MIO_PRESOLVED_NUMBINCONEVAR"

"MSK_IINF_MIO_PRESOLVED_NUMCONE"

"MSK_IINF_MIO_PRESOLVED_NUMCONEVAR"

"MSK_IINF_MIO_PRESOLVED_NUMCONTCONEVAR"

"MSK_IINF_MIO_PRESOLVED_NUMINTCONEVAR"

"MSK_IINF_MIO_PRESOLVED_NUMPEXPCONES"

"MSK_IINF_MIO_PRESOLVED_NUMQCONES"

"MSK_IINF_MIO_PRESOLVED_NUMRQCONES"

"MSK_IINF_PURIFY_DUAL_SUCCESS"

"MSK_IINF_PURIFY_PRIMAL_SUCCESS"

"MSK_LIINF_MIO_ANZ"

Removed

MSK_DATAFORMAT_XML

MSK_DINFITEM_MIO_HEURISTIC_TIME

MSK_DINFITEM_MIO_OPTIMIZER_TIME

MSK_IINFITEM_MIO_NEAR_ABSGAP_SATISFIED

MSK_IINFITEM_MIO_NEAR_RELGAP_SATISFIED

MSK_LIINFITEM_MIO_SIM_MAXITER_SETBACKS

MSK_MIONODESELTYPE_HYBRID

MSK_MIONODESELTYPE_WORST

MSK_PROBLEMTYPE_GECO

MSK_PROSTA_NEAR_DUAL_FEAS

MSK_PROSTA_NEAR_PRIM_AND_DUAL_FEAS

MSK_PROSTA_NEAR_PRIM_FEAS

MSK_SENSITIVITYTYPE_OPTIMAL_PARTITION

MSK_SOLSTA_NEAR_DUAL_FEAS

MSK_SOLSTA_NEAR_DUAL_INFEAS_CER

MSK_SOLSTA_NEAR_INTEGER_OPTIMAL

MSK_SOLSTA_NEAR_OPTIMAL

MSK_SOLSTA_NEAR_PRIM_AND_DUAL_FEAS

MSK_SOLSTA_NEAR_PRIM_FEAS

MSK_SOLSTA_NEAR_PRIM_INFEAS_CER

18.5 Response Codes

Added

"MSK_RES_ERR_APPENDING_TOO_BIG_CONE"

"MSK_RES_ERR_CBF_DUPLICATE_POW_CONES"

"MSK_RES_ERR_CBF_DUPLICATE_POW_STAR_CONES"

"MSK_RES_ERR_CBF_INVALID_DIMENSION_OF_CONES"

"MSK_RES_ERR_CBF_INVALID_EXP_DIMENSION"

"MSK_RES_ERR_CBF_INVALID_NUMBER_OF_CONES"

"MSK_RES_ERR_CBF_INVALID_POWER"

"MSK_RES_ERR_CBF_INVALID_POWER_CONE_INDEX"

"MSK_RES_ERR_CBF_INVALID_POWER_STAR_CONE_INDEX"

"MSK_RES_ERR_CBF_POWER_CONE_IS_TOO_LONG"

"MSK_RES_ERR_CBF_POWER_CONE_MISMATCH"

"MSK_RES_ERR_CBF_POWER_STAR_CONE_MISMATCH"

"MSK_RES_ERR_CBF_UNHANDLED_POWER_CONE_TYPE"

"MSK_RES_ERR_CBF_UNHANDLED_POWER_STAR_CONE_TYPE"

"MSK_RES_ERR_CONE_PARAMETER"

"MSK_RES_ERR_FORMAT_STRING"

"MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_CFIX"

"MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_FREE_CONSTRAINTS"

"MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_NONLINEAR"

"MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_RANGED_CONSTRAINTS"

"MSK_RES_ERR_NUM_ARGUMENTS"

"MSK_RES_ERR_PTF_FORMAT"

"MSK_RES_ERR_SHAPE_IS_TOO_LARGE"

"MSK_RES_ERR_SLICE_SIZE"

"MSK_RES_ERR_TOO_SMALL_A_TRUNCATION_VALUE"

"MSK_RES_WRN_EXP_CONES_WITH_VARIABLES_FIXED_AT_ZERO"

"MSK_RES_WRN_POW_CONES_WITH_ROOT_FIXED_AT_ZERO"

Removed

MSK_RES_ERR_CANNOT_CLONE_NL

MSK_RES_ERR_CANNOT_HANDLE_NL

MSK_RES_ERR_INVALID_ACCMODE

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_GENERAL_NL

MSK_RES_ERR_NONLINEAR_FUNCTIONS_NOT_ALLOWED

MSK_RES_ERR_NR_ARGUMENTS

MSK_RES_ERR_OPEN_DL

MSK_RES_ERR_USER_FUNC_RET

MSK_RES_ERR_USER_FUNC_RET_DATA

MSK_RES_ERR_USER_NLO_EVAL

MSK_RES_ERR_USER_NLO_EVAL_HESSUBI

MSK_RES_ERR_USER_NLO_EVAL_HESSUBJ

MSK_RES_ERR_USER_NLO_FUNC

MSK_RES_TRM_MIO_NEAR_ABS_GAP

MSK_RES_TRM_MIO_NEAR_REL_GAP

MSK_RES_WRN_CONSTRUCT_INVALID_SOL_ITG

MSK_RES_WRN_CONSTRUCT_NO_SOL_ITG

MSK_RES_WRN_CONSTRUCT_SOLUTION_INFEAS

MSK_RES_WRN_NO_NONLINEAR_FUNCTION_WRITE

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.

[BTN00] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Math. Programming*, 88(3):411–424, 2000.

[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.

[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.

[RTV97] C. Roos, T. Terlaky, and J. -Ph. Vial. *Theory and algorithms for linear optimization: an interior point approach*. John Wiley and Sons, New York, 1997.

[Wal00] S. W. Wallace. Decision making under uncertainty: is sensitivity of any use. *Oper. Res.*, 48(1):20–25, January 2000.

[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

Enumerations

basindtype, 218
"MSK_BI_RESERVERED", 219
"MSK_BI_NO_ERROR", 219
"MSK_BI_NEVER", 218
"MSK_BI_IF_FEASIBLE", 219
"MSK_BI_ALWAYS", 219
boundkey, 219
"MSK_BK_UP", 219
"MSK_BK_RA", 219
"MSK_BK_LO", 219
"MSK_BK_FX", 219
"MSK_BK_FR", 219
branchdir, 236
"MSK_BRANCH_DIR_UP", 236
"MSK_BRANCH_DIR_ROOT_LP", 236
"MSK_BRANCH_DIR_PSEUDOCOST", 236
"MSK_BRANCH_DIR_NEAR", 236
"MSK_BRANCH_DIR_GUIDED", 236
"MSK_BRANCH_DIR_FREE", 236
"MSK_BRANCH_DIR_FAR", 236
"MSK_BRANCH_DIR_DOWN", 236
callbackcode, 220
"MSK_CALLBACK_WRITE_OPF", 225
"MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX_BI", 225
"MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX", 225
"MSK_CALLBACK_UPDATE_PRIMAL_BI", 225
"MSK_CALLBACK_UPDATE_PRESOLVE", 225
"MSK_CALLBACK_UPDATE_DUAL_SIMPLEX_BI", 224
"MSK_CALLBACK_UPDATE_DUAL_SIMPLEX", 224
"MSK_CALLBACK_UPDATE_DUAL_BI", 224
"MSK_CALLBACK_SOLVING_REMOTE", 224
"MSK_CALLBACK_READ_OPF_SECTION", 224
"MSK_CALLBACK_READ_OPF", 224
"MSK_CALLBACK_PRIMAL_SIMPLEX", 224
"MSK_CALLBACK_NEW_INT_MIO", 224
"MSK_CALLBACK_INTPNT", 224
"MSK_CALLBACK_IM_SIMPLEX_BI", 224
"MSK_CALLBACK_IM_SIMPLEX", 224
"MSK_CALLBACK_IM_ROOT_CUTGEN", 224
"MSK_CALLBACK_IM_READ", 224
"MSK_CALLBACK_IM_QO_REFORMULATE", 224
"MSK_CALLBACK_IM_PRIMAL_SIMPLEX", 224
"MSK_CALLBACK_IM_PRIMAL_SENSIVITY", 224
"MSK_CALLBACK_IM_PRIMAL_BI", 224
"MSK_CALLBACK_IM_PRESOLVE", 224
"MSK_CALLBACK_IM_ORDER", 224
"MSK_CALLBACK_IM_MIO_PRIMAL_SIMPLEX", 224
"MSK_CALLBACK_IM_MIO_INTPNT", 223
"MSK_CALLBACK_IM_MIO_DUAL_SIMPLEX", 223
"MSK_CALLBACK_IM_MIO", 223
"MSK_CALLBACK_IM_LU", 223
"MSK_CALLBACK_IM_LICENSE_WAIT", 223
"MSK_CALLBACK_IM_INTPNT", 223
"MSK_CALLBACK_IM_FULL_CONVEXITY_CHECK", 223
"MSK_CALLBACK_IM_DUAL_SIMPLEX", 223
"MSK_CALLBACK_IM_DUAL_SENSIVITY", 223
"MSK_CALLBACK_IM_DUAL_BI", 223
"MSK_CALLBACK_IM_CONIC", 223
"MSK_CALLBACK_IM_BI", 223
"MSK_CALLBACK_END_WRITE", 223
"MSK_CALLBACK_END_TO_CONIC", 223
"MSK_CALLBACK_END_SIMPLEX_BI", 223
"MSK_CALLBACK_END_SIMPLEX", 223
"MSK_CALLBACK_END_ROOT_CUTGEN", 223
"MSK_CALLBACK_END_READ", 223
"MSK_CALLBACK_END_QCQO_REFORMULATE", 223
"MSK_CALLBACK_END_PRIMAL_SIMPLEX_BI", 223
"MSK_CALLBACK_END_PRIMAL_SIMPLEX", 223
"MSK_CALLBACK_END_PRIMAL_SETUP_BI", 222
"MSK_CALLBACK_END_PRIMAL_SENSITIVITY", 222
"MSK_CALLBACK_END_PRIMAL_REPAIR", 222
"MSK_CALLBACK_END_PRIMAL_BI", 222
"MSK_CALLBACK_END_PRESOLVE", 222
"MSK_CALLBACK_END_OPTIMIZER", 222
"MSK_CALLBACK_END_MIO", 222
"MSK_CALLBACK_END_LICENSE_WAIT", 222
"MSK_CALLBACK_END_INTPNT", 222
"MSK_CALLBACK_END_INFEAS_ANA", 222
"MSK_CALLBACK_END_FULL_CONVEXITY_CHECK", 222
"MSK_CALLBACK_END_DUAL_SIMPLEX_BI", 222
"MSK_CALLBACK_END_DUAL_SIMPLEX", 222
"MSK_CALLBACK_END_DUAL_SETUP_BI", 222
"MSK_CALLBACK_END_DUAL_SENSITIVITY", 222
"MSK_CALLBACK_END_DUAL_BI", 222
"MSK_CALLBACK_END_CONIC", 222
"MSK_CALLBACK_END_BI", 222
"MSK_CALLBACK_DUAL_SIMPLEX", 222
"MSK_CALLBACK_CONIC", 222
"MSK_CALLBACK_BEGIN_WRITE", 222
"MSK_CALLBACK_BEGIN_TO_CONIC", 222
"MSK_CALLBACK_BEGIN_SIMPLEX_BI", 222
"MSK_CALLBACK_BEGIN_SIMPLEX", 221
"MSK_CALLBACK_BEGIN_ROOT_CUTGEN", 221
"MSK_CALLBACK_BEGIN_READ", 221

"MSK_CALLBACK_BEGIN_QCQO_REFORMULATE", 221
"MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX_BI", 221
"MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX", 221
"MSK_CALLBACK_BEGIN_PRIMAL_SETUP_BI", 221
"MSK_CALLBACK_BEGIN_PRIMAL_SENSITIVITY", 221
"MSK_CALLBACK_BEGIN_PRIMAL_REPAIR", 221
"MSK_CALLBACK_BEGIN_PRIMAL_BI", 221
"MSK_CALLBACK_BEGIN_PRESOLVE", 221
"MSK_CALLBACK_BEGIN_OPTIMIZER", 221
"MSK_CALLBACK_BEGIN_MIO", 221
"MSK_CALLBACK_BEGIN_LICENSE_WAIT", 221
"MSK_CALLBACK_BEGIN_INTPNT", 221
"MSK_CALLBACK_BEGIN_INFEAS_ANA", 221
"MSK_CALLBACK_BEGIN_FULL_CONVEXITY_CHECK", 221
"MSK_CALLBACK_BEGIN_DUAL_SIMPLEX_BI", 221
"MSK_CALLBACK_BEGIN_DUAL_SIMPLEX", 221
"MSK_CALLBACK_BEGIN_DUAL_SETUP_BI", 221
"MSK_CALLBACK_BEGIN_DUAL_SENSITIVITY", 221
"MSK_CALLBACK_BEGIN_DUAL_BI", 221
"MSK_CALLBACK_BEGIN_CONIC", 221
"MSK_CALLBACK_BEGIN_BI", 220
checkconvexitytype, 225
"MSK_CHECK_CONVEXITY_SIMPLE", 225
"MSK_CHECK_CONVEXITY_NONE", 225
"MSK_CHECK_CONVEXITY_FULL", 225
compresstype, 225
"MSK_COMPRESS_ZSTD", 225
"MSK_COMPRESS_NONE", 225
"MSK_COMPRESS_GZIP", 225
"MSK_COMPRESS_FREE", 225
conetype, 225
"MSK_CT_ZERO", 225
"MSK_CT_RQUAD", 225
"MSK_CT_QUAD", 225
"MSK_CT_PPOW", 225
"MSK_CT_PEXP", 225
"MSK_CT_DPOW", 225
"MSK_CT_DEXP", 225
dataformat, 226
"MSK_DATA_FORMAT_TASK", 226
"MSK_DATA_FORMAT_PTF", 226
"MSK_DATA_FORMAT_OP", 226
"MSK_DATA_FORMAT_MPS", 226
"MSK_DATA_FORMAT_LP", 226
"MSK_DATA_FORMAT_JSON_TASK", 226
"MSK_DATA_FORMAT_FREE_MPS", 226
"MSK_DATA_FORMAT_EXTENSION", 226
"MSK_DATA_FORMAT_CB", 226
dinfitem, 226
"MSK_DINF_TO_CONIC_TIME", 231
"MSK_DINF_SOL_ITR_PVIOLVAR", 231
"MSK_DINF_SOL_ITR_PVIOLCONES", 231
"MSK_DINF_SOL_ITR_PVIOLCON", 231
"MSK_DINF_SOL_ITR_PVIOLBARVAR", 230
"MSK_DINF_SOL_ITR_PRIMAL_OBJ", 230
"MSK_DINF_SOL_ITR_NRM_Y", 230
"MSK_DINF_SOL_ITR_NRM_XX", 230
"MSK_DINF_SOL_ITR_NRM_XC", 230
"MSK_DINF_SOL_ITR_NRM_SUX", 230
"MSK_DINF_SOL_ITR_NRM_SUC", 230
"MSK_DINF_SOL_ITR_NRM_SNX", 230
"MSK_DINF_SOL_ITR_NRM_SLX", 230
"MSK_DINF_SOL_ITR_NRM_SLC", 230
"MSK_DINF_SOL_ITR_NRM_BARX", 230
"MSK_DINF_SOL_ITR_NRM_BARS", 230
"MSK_DINF_SOL_ITR_DVIOLVAR", 230
"MSK_DINF_SOL_ITR_DVIOLCONES", 230
"MSK_DINF_SOL_ITR_DVIOLCON", 230
"MSK_DINF_SOL_ITR_DVIOLBARVAR", 230
"MSK_DINF_SOL_ITR_DUAL_OBJ", 230
"MSK_DINF_SOL_ITG_PVIOLVAR", 230
"MSK_DINF_SOL_ITG_PVIOLITG", 230
"MSK_DINF_SOL_ITG_PVIOLCONES", 230
"MSK_DINF_SOL_ITG_PVIOLCON", 229
"MSK_DINF_SOL_ITG_PVIOLBARVAR", 229
"MSK_DINF_SOL_ITG_PRIMAL_OBJ", 229
"MSK_DINF_SOL_ITG_NRM_XX", 229
"MSK_DINF_SOL_ITG_NRM_XC", 229
"MSK_DINF_SOL_ITG_NRM_BARX", 229
"MSK_DINF_SOL_BAS_PVIOLVAR", 229
"MSK_DINF_SOL_BAS_PVIOLCON", 229
"MSK_DINF_SOL_BAS_PRIMAL_OBJ", 229
"MSK_DINF_SOL_BAS_NRM_Y", 229
"MSK_DINF_SOL_BAS_NRM_XX", 229
"MSK_DINF_SOL_BAS_NRM_XC", 229
"MSK_DINF_SOL_BAS_NRM_SUX", 229
"MSK_DINF_SOL_BAS_NRM_SUC", 229
"MSK_DINF_SOL_BAS_NRM_SLX", 229
"MSK_DINF_SOL_BAS_NRM_SLC", 229
"MSK_DINF_SOL_BAS_NRM_BARX", 229
"MSK_DINF_SOL_BAS_DVIOLVAR", 229
"MSK_DINF_SOL_BAS_DVIOLCON", 229
"MSK_DINF_SOL_BAS_DUAL_OBJ", 229
"MSK_DINF_SIM_TIME", 229
"MSK_DINF_SIM_PRIMAL_TIME", 228
"MSK_DINF_SIM_OBJ", 228
"MSK_DINF_SIM_FEAS", 228
"MSK_DINF_SIM_DUAL_TIME", 228
"MSK_DINF_RD_TIME", 228
"MSK_DINF_QCQO_REFORMULATE_WORST_CHOLESKY_DIAG_SCALING", 228
"MSK_DINF_QCQO_REFORMULATE_WORST_CHOLESKY_COLUMN_SCALING 228
"MSK_DINF_QCQO_REFORMULATE_TIME", 228
"MSK_DINF_QCQO_REFORMULATE_MAX_PERTURBATION", 228
"MSK_DINF_PRIMAL_REPAIR_PENALTY_OBJ", 228
"MSK_DINF_PRESOLVE_TIME", 228
"MSK_DINF_PRESOLVE_LINDEP_TIME", 228
"MSK_DINF_PRESOLVE_ELI_TIME", 228
"MSK_DINF_OPTIMIZER_TIME", 228
"MSK_DINF_MIO_USER_OBJ_CUT", 228
"MSK_DINF_MIO_TIME", 228
"MSK_DINF_MIO_ROOT_PRESOLVE_TIME", 228

"MSK_DINF_MIO_ROOT_OPTIMIZER_TIME", 228
"MSK_DINF_MIO_ROOT_CUTGEN_TIME", 228
"MSK_DINF_MIO_PROBING_TIME", 228
"MSK_DINF_MIO_OBJ_REL_GAP", 228
"MSK_DINF_MIO_OBJ_INT", 227
"MSK_DINF_MIO_OBJ_BOUND", 227
"MSK_DINF_MIO_OBJ_ABS_GAP", 227
"MSK_DINF_MIO_KNAPSACK_COVER_SEPARATION_TIME", 227
"MSK_DINF_MIO_IMPLIED_BOUND_TIME", 227
"MSK_DINF_MIO_GMI_SEPARATION_TIME", 227
"MSK_DINF_MIO_DUAL_BOUND_AFTER_PRESOLVE", 227
"MSK_DINF_MIO_CONSTRUCT_SOLUTION_OBJ", 227
"MSK_DINF_MIO_CMIR_SEPARATION_TIME", 227
"MSK_DINF_MIO_CLIQUE_SEPARATION_TIME", 227
"MSK_DINF_INTPNT_TIME", 227
"MSK_DINF_INTPNT_PRIMAL_OBJ", 227
"MSK_DINF_INTPNT_PRIMAL_FEAS", 227
"MSK_DINF_INTPNT_ORDER_TIME", 227
"MSK_DINF_INTPNT_OPT_STATUS", 227
"MSK_DINF_INTPNT_FACTOR_NUM_FLOPS", 227
"MSK_DINF_INTPNT_DUAL_OBJ", 227
"MSK_DINF_INTPNT_DUAL_FEAS", 227
"MSK_DINF_BI_TIME", 227
"MSK_DINF_BI_PRIMAL_TIME", 227
"MSK_DINF_BI_DUAL_TIME", 226
"MSK_DINF_BI_CLEAN_TIME", 226
"MSK_DINF_BI_CLEAN_PRIMAL_TIME", 226
"MSK_DINF_BI_CLEAN_DUAL_TIME", 226
dparam, 163
feature, 231
"MSK_FEATURE_PTS", 231
"MSK_FEATURE_PTON", 231
iinfitem, 231
"MSK_IINF_STO_NUM_A_REALLOC", 235
"MSK_IINF_SOL_ITR_SOLSTA", 235
"MSK_IINF_SOL_ITR_PROSTA", 235
"MSK_IINF_SOL_ITG_SOLSTA", 235
"MSK_IINF_SOL_ITG_PROSTA", 235
"MSK_IINF_SOL_BAS_SOLSTA", 235
"MSK_IINF_SOL_BAS_PROSTA", 235
"MSK_IINF_SIM_SOLVE_DUAL", 235
"MSK_IINF_SIM_PRIMAL_ITER", 235
"MSK_IINF_SIM_PRIMAL_INF_ITER", 235
"MSK_IINF_SIM_PRIMAL_HOTSTART_LU", 235
"MSK_IINF_SIM_PRIMAL_HOTSTART", 235
"MSK_IINF_SIM_PRIMAL_DEG_ITER", 235
"MSK_IINF_SIM_NUMVAR", 235
"MSK_IINF_SIM_NUMCON", 235
"MSK_IINF_SIM_DUAL_ITER", 235
"MSK_IINF_SIM_DUAL_INF_ITER", 235
"MSK_IINF_SIM_DUAL_HOTSTART_LU", 235
"MSK_IINF_SIM_DUAL_HOTSTART", 235
"MSK_IINF_SIM_DUAL_DEG_ITER", 235
"MSK_IINF_RD_PROTYPE", 235
"MSK_IINF_RD_NUMVAR", 235
"MSK_IINF_RD_NUMQ", 235
"MSK_IINF_RD_NUMINTVAR", 235
"MSK_IINF_RD_NUMCONE", 234
"MSK_IINF_RD_NUMCON", 234
"MSK_IINF_RD_NUMBARVAR", 234
"MSK_IINF_PURIFY_PRIMAL_SUCCESS", 234
"MSK_IINF_PURIFY_DUAL_SUCCESS", 234
"MSK_IINF_OPTIMIZE_RESPONSE", 234
"MSK_IINF_OPT_NUMVAR", 234
"MSK_IINF_OPT_NUMCON", 234
"MSK_IINF_MIO_USER_OBJ_CUT", 234
"MSK_IINF_MIO_TOTAL_NUM_CUTS", 234
"MSK_IINF_MIO_RELGAP_SATISFIED", 234
"MSK_IINF_MIO_PRESOLVED_NUMVAR", 234
"MSK_IINF_MIO_PRESOLVED_NUMRQCONES", 234
"MSK_IINF_MIO_PRESOLVED_NUMQCONES", 234
"MSK_IINF_MIO_PRESOLVED_NUMPEXPCONES", 234
"MSK_IINF_MIO_PRESOLVED_NUMINTCONEVAR", 234
"MSK_IINF_MIO_PRESOLVED_NUMINT", 234
"MSK_IINF_MIO_PRESOLVED_NUMCONTCONEVAR", 234
"MSK_IINF_MIO_PRESOLVED_NUMCONT", 234
"MSK_IINF_MIO_PRESOLVED_NUMCONEVAR", 234
"MSK_IINF_MIO_PRESOLVED_NUMCONE", 234
"MSK_IINF_MIO_PRESOLVED_NUMCON", 234
"MSK_IINF_MIO_PRESOLVED_NUMBINCONEVAR", 234
"MSK_IINF_MIO_PRESOLVED_NUMBIN", 234
"MSK_IINF_MIO_OBJ_BOUND_DEFINED", 233
"MSK_IINF_MIO_NUMVAR", 233
"MSK_IINF_MIO_NUMRQCONES", 233
"MSK_IINF_MIO_NUMQCONES", 233
"MSK_IINF_MIO_NUMPEXPCONES", 233
"MSK_IINF_MIO_NUMINTCONEVAR", 233
"MSK_IINF_MIO_NUMINT", 233
"MSK_IINF_MIO_NUMCONTCONEVAR", 233
"MSK_IINF_MIO_NUMCONT", 233
"MSK_IINF_MIO_NUMCONEVAR", 233
"MSK_IINF_MIO_NUMCONE", 233
"MSK_IINF_MIO_NUMCON", 233
"MSK_IINF_MIO_NUMBINCONEVAR", 233
"MSK_IINF_MIO_NUMBIN", 233
"MSK_IINF_MIO_NUM_REPEATED_PRESOLVE", 233
"MSK_IINF_MIO_NUM_RELAX", 233
"MSK_IINF_MIO_NUM_KNAPSACK_COVER_CUTS", 233
"MSK_IINF_MIO_NUM_INT_SOLUTIONS", 233
"MSK_IINF_MIO_NUM_IMPLIED_BOUND_CUTS", 233
"MSK_IINF_MIO_NUM_GOMORY_CUTS", 233
"MSK_IINF_MIO_NUM_CMIR_CUTS", 233
"MSK_IINF_MIO_NUM_CLIQUE_CUTS", 233
"MSK_IINF_MIO_NUM_BRANCH", 233
"MSK_IINF_MIO_NUM_ACTIVE_NODES", 233
"MSK_IINF_MIO_NODE_DEPTH", 232
"MSK_IINF_MIO_INITIAL_SOLUTION", 232
"MSK_IINF_MIO_CONSTRUCT_SOLUTION", 232
"MSK_IINF_MIO_CONSTRUCT_NUM_ROUNDINGS", 232
"MSK_IINF_MIO_CLIQUE_TABLE_SIZE", 232
"MSK_IINF_MIO_ABSGAP_SATISFIED", 232
"MSK_IINF_INTPNT_SOLVE_DUAL", 232
"MSK_IINF_INTPNT_NUM_THREADS", 232

"MSK_IINF_INTPNT_ITER", 232
"MSK_IINF_INTPNT_FACTOR_DIM_DENSE", 232
"MSK_IINF_ANA_PRO_NUM_VAR_UP", 232
"MSK_IINF_ANA_PRO_NUM_VAR_RA", 232
"MSK_IINF_ANA_PRO_NUM_VAR_LO", 232
"MSK_IINF_ANA_PRO_NUM_VAR_INT", 232
"MSK_IINF_ANA_PRO_NUM_VAR_FR", 232
"MSK_IINF_ANA_PRO_NUM_VAR_EQ", 232
"MSK_IINF_ANA_PRO_NUM_VAR_CONT", 232
"MSK_IINF_ANA_PRO_NUM_VAR_BIN", 232
"MSK_IINF_ANA_PRO_NUM_VAR", 232
"MSK_IINF_ANA_PRO_NUM_CON_UP", 232
"MSK_IINF_ANA_PRO_NUM_CON_RA", 232
"MSK_IINF_ANA_PRO_NUM_CON_LO", 232
"MSK_IINF_ANA_PRO_NUM_CON_FR", 232
"MSK_IINF_ANA_PRO_NUM_CON_EQ", 232
"MSK_IINF_ANA_PRO_NUM_CON", 231
inftype, 236
"MSK_INF_LINT_TYPE", 236
"MSK_INF_INT_TYPE", 236
"MSK_INF_DOU_TYPE", 236
intpnthotstart, 220
"MSK_INTPNT_HOTSTART_PRIMAL_DUAL", 220
"MSK_INTPNT_HOTSTART_PRIMAL", 220
"MSK_INTPNT_HOTSTART_NONE", 220
"MSK_INTPNT_HOTSTART_DUAL", 220
iomode, 236
"MSK_IOMODE_WRITE", 236
"MSK_IOMODE_READWRITE", 236
"MSK_IOMODE_READ", 236
iparam, 171
liinfitem, 231
"MSK_LIINF_RD_NUMQNZ", 231
"MSK_LIINF_RD_NUMANZ", 231
"MSK_LIINF_MIO_SIMPLEX_ITER", 231
"MSK_LIINF_MIO_PRESOLVED_ANZ", 231
"MSK_LIINF_MIO_INTPNT_ITER", 231
"MSK_LIINF_MIO_ANZ", 231
"MSK_LIINF_INTPNT_FACTOR_NUM_NZ", 231
"MSK_LIINF_BI_PRIMAL_ITER", 231
"MSK_LIINF_BI_DUAL_ITER", 231
"MSK_LIINF_BI_CLEAN_PRIMAL_ITER", 231
"MSK_LIINF_BI_CLEAN_PRIMAL_DEG_ITER", 231
"MSK_LIINF_BI_CLEAN_DUAL_ITER", 231
"MSK_LIINF_BI_CLEAN_DUAL_DEG_ITER", 231
mark, 219
"MSK_MARK_UP", 219
"MSK_MARK_LO", 219
miocontsoltype, 236
"MSK_MIO_CONT_SOL_ROOT", 236
"MSK_MIO_CONT_SOL_NONE", 236
"MSK_MIO_CONT_SOL_ITG_REL", 236
"MSK_MIO_CONT_SOL_ITG", 236
miomode, 237
"MSK_MIO_MODE_SATISFIED", 237
"MSK_MIO_MODE_IGNORED", 237
mionodeseltype, 237
"MSK_MIO_NODE_SELECTION_PSEUDO", 237
"MSK_MIO_NODE_SELECTION_FREE", 237
"MSK_MIO_NODE_SELECTION_FIRST", 237
"MSK_MIO_NODE_SELECTION_BEST", 237
mpsformat, 237
"MSK_MPS_FORMAT_STRICT", 237
"MSK_MPS_FORMAT_RELAXED", 237
"MSK_MPS_FORMAT_FREE", 237
"MSK_MPS_FORMAT_CPLEX", 237
nametype, 225
"MSK_NAME_TYPE_MPS", 225
"MSK_NAME_TYPE_LP", 226
"MSK_NAME_TYPE_GEN", 225
objsense, 237
"MSK_OBJECTIVE_SENSE_MINIMIZE", 237
"MSK_OBJECTIVE_SENSE_MAXIMIZE", 237
onoffkey, 237
"MSK_ON", 237
"MSK_OFF", 237
optimizertype, 237
"MSK_OPTIMIZER_PRIMAL_SIMPLEX", 238
"MSK_OPTIMIZER_MIXED_INT", 238
"MSK_OPTIMIZER_INTPNT", 237
"MSK_OPTIMIZER_FREE_SIMPLEX", 237
"MSK_OPTIMIZER_FREE", 237
"MSK_OPTIMIZER_DUAL_SIMPLEX", 237
"MSK_OPTIMIZER_CONIC", 237
orderingtype, 238
"MSK_ORDER_METHOD_TRY_GRAPHPAR", 238
"MSK_ORDER_METHOD_NONE", 238
"MSK_ORDER_METHOD_FREE", 238
"MSK_ORDER_METHOD_FORCE_GRAPHPAR", 238
"MSK_ORDER_METHOD_EXPERIMENTAL", 238
"MSK_ORDER_METHOD_APPMINLOC", 238
parametertype, 238
"MSK_PAR_STR_TYPE", 238
"MSK_PAR_INVALID_TYPE", 238
"MSK_PAR_INT_TYPE", 238
"MSK_PAR_DOU_TYPE", 238
presolvemode, 238
"MSK_PRESOLVE_MODE_ON", 238
"MSK_PRESOLVE_MODE_OFF", 238
"MSK_PRESOLVE_MODE_FREE", 238
problemitem, 238
"MSK_PI_VAR", 238
"MSK_PI_CONE", 238
"MSK_PI_CON", 238
problemtype, 238
"MSK_PROBTYPE_QO", 238
"MSK_PROBTYPE_QCQO", 239
"MSK_PROBTYPE_MIXED", 239
"MSK_PROBTYPE_LO", 238
"MSK_PROBTYPE_CONIC", 239
prosta, 239
"MSK_PRO_STA_UNKNOWN", 239
"MSK_PRO_STA_PRIM_INFEAS_OR_UNBOUNDED", 239
"MSK_PRO_STA_PRIM_INFEAS", 239
"MSK_PRO_STA_PRIM_FEAS", 239
"MSK_PRO_STA_PRIM_AND_DUAL_INFEAS", 239

"MSK_PRO_STA_PRIM_AND_DUAL_FEAS", 239
"MSK_PRO_STA_ILL_POSED", 239
"MSK_PRO_STA_DUAL_INFEAS", 239
"MSK_PRO_STA_DUAL_FEAS", 239
purify, 220
"MSK_PURIFY_PRIMAL_DUAL", 220
"MSK_PURIFY_PRIMAL", 220
"MSK_PURIFY_NONE", 220
"MSK_PURIFY_DUAL", 220
"MSK_PURIFY_AUTO", 220
rescode, 201
rescodetype, 239
"MSK_RESPONSE_WRN", 239
"MSK_RESPONSE_UNK", 239
"MSK_RESPONSE_TRM", 239
"MSK_RESPONSE_OK", 239
"MSK_RESPONSE_ERR", 239
scalingmethod, 240
"MSK_SCALING_METHOD_POW2", 240
"MSK_SCALING_METHOD_FREE", 240
scalingtype, 239
"MSK_SCALING_NONE", 240
"MSK_SCALING_MODERATE", 240
"MSK_SCALING_FREE", 239
"MSK_SCALING_AGGRESSIVE", 240
scopr, 226
"MSK_OPR_SQRT", 226
"MSK_OPR_POW", 226
"MSK_OPR_LOG", 226
"MSK_OPR_EXP", 226
"MSK_OPR_ENT", 226
sensitivitytype, 240
"MSK_SENSITIVITY_TYPE_BASIS", 240
simdegen, 219
"MSK_SIM_DEGEN_NONE", 219
"MSK_SIM_DEGEN_MODERATE", 219
"MSK_SIM_DEGEN_MINIMUM", 219
"MSK_SIM_DEGEN_FREE", 219
"MSK_SIM_DEGEN_AGGRESSIVE", 219
simdupvec, 220
"MSK_SIM_EXPLOIT_DUPVEC_ON", 220
"MSK_SIM_EXPLOIT_DUPVEC_OFF", 220
"MSK_SIM_EXPLOIT_DUPVEC_FREE", 220
simhotstart, 220
"MSK_SIM_HOTSTART_STATUS_KEYS", 220
"MSK_SIM_HOTSTART_NONE", 220
"MSK_SIM_HOTSTART_FREE", 220
simreform, 220
"MSK_SIM_REFORMULATION_ON", 220
"MSK_SIM_REFORMULATION_OFF", 220
"MSK_SIM_REFORMULATION_FREE", 220
"MSK_SIM_REFORMULATION_AGGRESSIVE", 220
simseltype, 240
"MSK_SIM_SELECTION_SE", 240
"MSK_SIM_SELECTION_PARTIAL", 240
"MSK_SIM_SELECTION_FULL", 240
"MSK_SIM_SELECTION_FREE", 240
"MSK_SIM_SELECTION_DEVEX", 240
"MSK_SIM_SELECTION_ASE", 240
solitem, 240
"MSK_SOL_ITEM_Y", 240
"MSK_SOL_ITEM_XX", 240
"MSK_SOL_ITEM_XC", 240
"MSK_SOL_ITEM_SUX", 240
"MSK_SOL_ITEM_SUC", 240
"MSK_SOL_ITEM_SNX", 240
"MSK_SOL_ITEM_SLX", 240
"MSK_SOL_ITEM_SLC", 240
solsta, 241
"MSK_SOL_STA_UNKNOWN", 241
"MSK_SOL_STA_PRIM_INFEAS_CER", 241
"MSK_SOL_STA_PRIM_ILLPOSED_CER", 241
"MSK_SOL_STA_PRIM_FEAS", 241
"MSK_SOL_STA_PRIM_AND_DUAL_FEAS", 241
"MSK_SOL_STA_OPTIMAL", 241
"MSK_SOL_STA_INTEGER_OPTIMAL", 241
"MSK_SOL_STA_DUAL_INFEAS_CER", 241
"MSK_SOL_STA_DUAL_ILLPOSED_CER", 241
"MSK_SOL_STA_DUAL_FEAS", 241
soltype, 241
"MSK_SOL_ITR", 241
"MSK_SOL_ITG", 241
"MSK_SOL_BAS", 241
solveform, 241
"MSK_SOLVE_PRIMAL", 241
"MSK_SOLVE_FREE", 241
"MSK_SOLVE_DUAL", 241
sparam, 198
stakey, 241
"MSK_SK_UPR", 241
"MSK_SK_UNK", 241
"MSK_SK_SUPBAS", 241
"MSK_SK_LOW", 241
"MSK_SK_INF", 242
"MSK_SK_FIX", 242
"MSK_SK_BAS", 241
startpointtype, 242
"MSK_STARTING_POINT_SATISFY_BOUNDS", 242
"MSK_STARTING_POINT_GUESS", 242
"MSK_STARTING_POINT_FREE", 242
"MSK_STARTING_POINT_CONSTANT", 242
streamtype, 242
"MSK_STREAM_WRN", 242
"MSK_STREAM_MSG", 242
"MSK_STREAM_LOG", 242
"MSK_STREAM_ERR", 242
symmattype, 226
"MSK_SYMMAT_TYPE_SPARSE", 226
transpose, 219
"MSK_TRANSPOSE_YES", 219
"MSK_TRANSPOSE_NO", 219
uplo, 219
"MSK_UPLO_UP", 219
"MSK_UPLO_LO", 219
value, 242
"MSK_MAX_STR_LEN", 242

"MSK_LICENSE_BUFFER_LENGTH", 242
variabletype, 242
"MSK_VAR_TYPE_INT", 242
"MSK_VAR_TYPE_CONT", 242
xmlwriteroutputtype, 239
"MSK_WRITE_XML_MODE_ROW", 239
"MSK_WRITE_XML_MODE_COL", 239

Functions

intlinprog, 139
linprog, 139
lsqlin, 140
lsqnonneg, 141
mosekopt, 136
msklpopt, 137
mskoptimget, 138
mskoptimset, 138
mskqpopt, 137
quadprog, 142

Parameters

Double parameters, 163
MSK_DPAR_ANA_SOL_INFEAS_TOL, 163
MSK_DPAR_BASIS_REL_TOL_S, 163
MSK_DPAR_BASIS_TOL_S, 163
MSK_DPAR_BASIS_TOL_X, 163
MSK_DPAR_CHECK_CONVEXITY_REL_TOL, 163
MSK_DPAR_DATA_SYM_MAT_TOL, 163
MSK_DPAR_DATA_SYM_MAT_TOL_HUGE, 164
MSK_DPAR_DATA_SYM_MAT_TOL_LARGE, 164
MSK_DPAR_DATA_TOL_AIJ_HUGE, 164
MSK_DPAR_DATA_TOL_AIJ_LARGE, 164
MSK_DPAR_DATA_TOL_BOUND_INF, 164
MSK_DPAR_DATA_TOL_BOUND_WRN, 164
MSK_DPAR_DATA_TOL_C_HUGE, 164
MSK_DPAR_DATA_TOL_CJ_LARGE, 165
MSK_DPAR_DATA_TOL_QIJ, 165
MSK_DPAR_DATA_TOL_X, 165
MSK_DPAR_INTPNT_CO_TOL_DFEAS, 165
MSK_DPAR_INTPNT_CO_TOL_INFEAS, 165
MSK_DPAR_INTPNT_CO_TOL_MU_RED, 165
MSK_DPAR_INTPNT_CO_TOL_NEAR_REL, 165
MSK_DPAR_INTPNT_CO_TOL_PFEAS, 166
MSK_DPAR_INTPNT_CO_TOL_REL_GAP, 166
MSK_DPAR_INTPNT_QO_TOL_DFEAS, 166
MSK_DPAR_INTPNT_QO_TOL_INFEAS, 166
MSK_DPAR_INTPNT_QO_TOL_MU_RED, 166
MSK_DPAR_INTPNT_QO_TOL_NEAR_REL, 166
MSK_DPAR_INTPNT_QO_TOL_PFEAS, 167
MSK_DPAR_INTPNT_QO_TOL_REL_GAP, 167
MSK_DPAR_INTPNT_TOL_DFEAS, 167
MSK_DPAR_INTPNT_TOL_DSAFE, 167
MSK_DPAR_INTPNT_TOL_INFEAS, 167
MSK_DPAR_INTPNT_TOL_MU_RED, 167
MSK_DPAR_INTPNT_TOL_PATH, 167
MSK_DPAR_INTPNT_TOL_PFEAS, 168
MSK_DPAR_INTPNT_TOL_PSAFE, 168
MSK_DPAR_INTPNT_TOL_REL_GAP, 168
MSK_DPAR_INTPNT_TOL_REL_STEP, 168
MSK_DPAR_INTPNT_TOL_STEP_SIZE, 168
MSK_DPAR_LOWER_OBJ_CUT, 168
MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH, 169
MSK_DPAR_MIO_MAX_TIME, 169
MSK_DPAR_MIO_REL_GAP_CONST, 169
MSK_DPAR_MIO_TOL_ABS_GAP, 169
MSK_DPAR_MIO_TOL_ABS_RELAX_INT, 169
MSK_DPAR_MIO_TOL_FEAS, 169
MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT, 169
MSK_DPAR_MIO_TOL_REL_GAP, 169
MSK_DPAR_OPTIMIZER_MAX_TIME, 170
MSK_DPAR_PRESOLVE_TOL_ABS_LINDEP, 170
MSK_DPAR_PRESOLVE_TOL_AIJ, 170
MSK_DPAR_PRESOLVE_TOL_REL_LINDEP, 170
MSK_DPAR_PRESOLVE_TOL_S, 170
MSK_DPAR_PRESOLVE_TOL_X, 170
MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL, 170
MSK_DPAR_SEMIDEFINITE_TOL_APPROX, 171
MSK_DPAR_SIM_LU_TOL_REL_PIV, 171
MSK_DPAR_SIMPLEX_ABS_TOL_PIV, 171
MSK_DPAR_UPPER_OBJ_CUT, 171
MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH, 171
Integer parameters, 171
MSK_IPAR_ANA_SOL_BASIS, 171
MSK_IPAR_ANA_SOL_PRINT_VIOLATED, 172
MSK_IPAR_AUTO_SORT_A_BEFORE_OPT, 172
MSK_IPAR_AUTO_UPDATE_SOL_INFO, 172
MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE, 172
MSK_IPAR_BI_CLEAN_OPTIMIZER, 172
MSK_IPAR_BI_IGNORE_MAX_ITER, 172
MSK_IPAR_BI_IGNORE_NUM_ERROR, 172
MSK_IPAR_BI_MAX_ITERATIONS, 173
MSK_IPAR_CACHE_LICENSE, 173
MSK_IPAR_CHECK_CONVEXITY, 173
MSK_IPAR_COMPRESS_STATFILE, 173
MSK_IPAR_INFEAS_GENERIC_NAMES, 173
MSK_IPAR_INFEAS_PREFER_PRIMAL, 173
MSK_IPAR_INFEAS_REPORT_AUTO, 174
MSK_IPAR_INFEAS_REPORT_LEVEL, 174
MSK_IPAR_INTPNT_BASIS, 174
MSK_IPAR_INTPNT_DIFF_STEP, 174
MSK_IPAR_INTPNT_HOTSTART, 174
MSK_IPAR_INTPNT_MAX_ITERATIONS, 174
MSK_IPAR_INTPNT_MAX_NUM_COR, 174
MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS, 175
MSK_IPAR_INTPNT_MULTI_THREAD, 175
MSK_IPAR_INTPNT_OFF_COL_TRH, 175
MSK_IPAR_INTPNT_ORDER_METHOD, 175
MSK_IPAR_INTPNT_PURIFY, 175
MSK_IPAR_INTPNT_REGULARIZATION_USE, 175
MSK_IPAR_INTPNT_SCALING, 176
MSK_IPAR_INTPNT_SOLVE_FORM, 176
MSK_IPAR_INTPNT_STARTING_POINT, 176
MSK_IPAR_LICENSE_DEBUG, 176
MSK_IPAR_LICENSE_PAUSE_TIME, 176

MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS, 176
MSK_IPAR_LICENSE_TRH_EXPIRY_WRN, 176
MSK_IPAR_LICENSE_WAIT, 177
MSK_IPAR_LOG, 177
MSK_IPAR_LOG_ANA_PRO, 177
MSK_IPAR_LOG_BI, 177
MSK_IPAR_LOG_BI_FREQ, 177
MSK_IPAR_LOG_CHECK_CONVEXITY, 177
MSK_IPAR_LOG_CUT_SECOND_OPT, 178
MSK_IPAR_LOG_EXPAND, 178
MSK_IPAR_LOG_FEAS_REPAIR, 178
MSK_IPAR_LOG_FILE, 178
MSK_IPAR_LOG_INFEAS_ANA, 178
MSK_IPAR_LOG_INTPNT, 178
MSK_IPAR_LOG_MIO, 178
MSK_IPAR_LOG_MIO_FREQ, 179
MSK_IPAR_LOG_ORDER, 179
MSK_IPAR_LOG_PRESOLVE, 179
MSK_IPAR_LOG_RESPONSE, 179
MSK_IPAR_LOG_SENSITIVITY, 179
MSK_IPAR_LOG_SENSITIVITY_OPT, 179
MSK_IPAR_LOG_SIM, 180
MSK_IPAR_LOG_SIM_FREQ, 180
MSK_IPAR_LOG_SIM_MINOR, 180
MSK_IPAR_LOG_STORAGE, 180
MSK_IPAR_MAX_NUM_WARNINGS, 180
MSK_IPAR_MIO_BRANCH_DIR, 180
MSK_IPAR_MIO_CONIC_OUTER_APPROXIMATION, 180
MSK_IPAR_MIO_CUT_CLIQUE, 181
MSK_IPAR_MIO_CUT_CMIR, 181
MSK_IPAR_MIO_CUT_GMI, 181
MSK_IPAR_MIO_CUT_IMPLIED_BOUND, 181
MSK_IPAR_MIO_CUT_KNAPSACK_COVER, 181
MSK_IPAR_MIO_CUT_SELECTION_LEVEL, 181
MSK_IPAR_MIO_HEURISTIC_LEVEL, 181
MSK_IPAR_MIO_MAX_NUM_BRANCHES, 182
MSK_IPAR_MIO_MAX_NUM_RELAXS, 182
MSK_IPAR_MIO_MAX_NUM_SOLUTIONS, 182
MSK_IPAR_MIO_MODE, 182
MSK_IPAR_MIO_NODE_OPTIMIZER, 182
MSK_IPAR_MIO_NODE_SELECTION, 182
MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE, 183
MSK_IPAR_MIO_PROBING_LEVEL, 183
MSK_IPAR_MIO_PROPAGATE_OBJECTIVE_CONSTRAINT, 183
MSK_IPAR_MIO_RINS_MAX_NODES, 183
MSK_IPAR_MIO_ROOT_OPTIMIZER, 183
MSK_IPAR_MIO_ROOT_REPEAT_PRESOLVE_LEVEL, 183
MSK_IPAR_MIO_SEED, 184
MSK_IPAR_MIO_VB_DETECTION_LEVEL, 184
MSK_IPAR_MT_SPINCOUNT, 184
MSK_IPAR_NUM_THREADS, 184
MSK_IPAR_OPF_WRITE_HEADER, 184
MSK_IPAR_OPF_WRITE_HINTS, 184
MSK_IPAR_OPF_WRITE_LINE_LENGTH, 185
MSK_IPAR_OPF_WRITE_PARAMETERS, 185
MSK_IPAR_OPF_WRITE_PROBLEM, 185
MSK_IPAR_OPF_WRITE_SOL_BAS, 185
MSK_IPAR_OPF_WRITE_SOL_ITG, 185
MSK_IPAR_OPF_WRITE_SOL_ITR, 185
MSK_IPAR_OPF_WRITE_SOLUTIONS, 185
MSK_IPAR_OPTIMIZER, 186
MSK_IPAR_PARAM_READ_CASE_NAME, 186
MSK_IPAR_PARAM_READ_IGN_ERROR, 186
MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_FILL, 186
MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES, 186
MSK_IPAR_PRESOLVE_LEVEL, 186
MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH, 186
MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH, 187
MSK_IPAR_PRESOLVE_LINDEP_USE, 187
MSK_IPAR_PRESOLVE_MAX_NUM_PASS, 187
MSK_IPAR_PRESOLVE_MAX_NUM_REDUCTIONS, 187
MSK_IPAR_PRESOLVE_USE, 187
MSK_IPAR_PRIMAL_REPAIR_OPTIMIZER, 187
MSK_IPAR_READ_DEBUG, 188
MSK_IPAR_READ_KEEP_FREE_CON, 188
MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU, 188
MSK_IPAR_READ_LP_QUOTED_NAMES, 188
MSK_IPAR_READ_MPS_FORMAT, 188
MSK_IPAR_READ_MPS_WIDTH, 188
MSK_IPAR_READ_TASK_IGNORE_PARAM, 188
MSK_IPAR_REMOVE_UNUSED_SOLUTIONS, 189
MSK_IPAR_SENSITIVITY_ALL, 189
MSK_IPAR_SENSITIVITY_OPTIMIZER, 189
MSK_IPAR_SENSITIVITY_TYPE, 189
MSK_IPAR_SIM_BASIS_FACTOR_USE, 189
MSK_IPAR_SIM_DEGEN, 189
MSK_IPAR_SIM_DUAL_CRASH, 189
MSK_IPAR_SIM_DUAL_PHASEONE_METHOD, 190
MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION, 190
MSK_IPAR_SIM_DUAL_SELECTION, 190
MSK_IPAR_SIM_EXPLOIT_DUPVEC, 190
MSK_IPAR_SIM_HOTSTART, 190
MSK_IPAR_SIM_HOTSTART_LU, 190
MSK_IPAR_SIM_MAX_ITERATIONS, 190
MSK_IPAR_SIM_MAX_NUM_SETBACKS, 191
MSK_IPAR_SIM_NON_SINGULAR, 191
MSK_IPAR_SIM_PRIMAL_CRASH, 191
MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD, 191
MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION, 191
MSK_IPAR_SIM_PRIMAL_SELECTION, 191
MSK_IPAR_SIM_REFACTOR_FREQ, 192
MSK_IPAR_SIM_REFORMULATION, 192
MSK_IPAR_SIM_SAVE_LU, 192
MSK_IPAR_SIM_SCALING, 192
MSK_IPAR_SIM_SCALING_METHOD, 192
MSK_IPAR_SIM_SEED, 192
MSK_IPAR_SIM_SOLVE_FORM, 192
MSK_IPAR_SIM_STABILITY_PRIORITY, 193
MSK_IPAR_SIM_SWITCH_OPTIMIZER, 193
MSK_IPAR_SOL_FILTER_KEEP_BASIC, 193
MSK_IPAR_SOL_FILTER_KEEP_RANGED, 193
MSK_IPAR_SOL_READ_NAME_WIDTH, 193
MSK_IPAR_SOL_READ_WIDTH, 193

MSK_IPAR_SOLUTION_CALLBACK, 193
MSK_IPAR_TIMING_LEVEL, 194
MSK_IPAR_WRITE_BAS_CONSTRAINTS, 194
MSK_IPAR_WRITE_BAS_HEAD, 194
MSK_IPAR_WRITE_BAS_VARIABLES, 194
MSK_IPAR_WRITE_COMPRESSION, 194
MSK_IPAR_WRITE_DATA_PARAM, 194
MSK_IPAR_WRITE_FREE_CON, 194
MSK_IPAR_WRITE_GENERIC_NAMES, 195
MSK_IPAR_WRITE_GENERIC_NAMES_IO, 195
MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_ITEMS, 195
MSK_IPAR_WRITE_INT_CONSTRAINTS, 195
MSK_IPAR_WRITE_INT_HEAD, 195
MSK_IPAR_WRITE_INT_VARIABLES, 195
MSK_IPAR_WRITE_LP_FULL_OBJ, 195
MSK_IPAR_WRITE_LP_LINE_WIDTH, 196
MSK_IPAR_WRITE_LP_QUOTED_NAMES, 196
MSK_IPAR_WRITE_LP_STRICT_FORMAT, 196
MSK_IPAR_WRITE_LP_TERMS_PER_LINE, 196
MSK_IPAR_WRITE_MPS_FORMAT, 196
MSK_IPAR_WRITE_MPS_INT, 196
MSK_IPAR_WRITE_PRECISION, 196
MSK_IPAR_WRITE_SOL_BARVARIABLES, 197
MSK_IPAR_WRITE_SOL_CONSTRAINTS, 197
MSK_IPAR_WRITE_SOL_HEAD, 197
MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES, 197
MSK_IPAR_WRITE_SOL_VARIABLES, 197
MSK_IPAR_WRITE_TASK_INC_SOL, 197
MSK_IPAR_WRITE_XML_MODE, 197
String parameters, 198
MSK_SPAR_BAS_SOL_FILE_NAME, 198
MSK_SPAR_DATA_FILE_NAME, 198
MSK_SPAR_DEBUG_FILE_NAME, 198
MSK_SPAR_INT_SOL_FILE_NAME, 198
MSK_SPAR_ITR_SOL_FILE_NAME, 198
MSK_SPAR_MIO_DEBUG_STRING, 198
MSK_SPAR_PARAM_COMMENT_SIGN, 198
MSK_SPAR_PARAM_READ_FILE_NAME, 198
MSK_SPAR_PARAM_WRITE_FILE_NAME, 199
MSK_SPAR_READ_MPS_BOU_NAME, 199
MSK_SPAR_READ_MPS_OBJ_NAME, 199
MSK_SPAR_READ_MPS_RAN_NAME, 199
MSK_SPAR_READ_MPS_RHS_NAME, 199
MSK_SPAR_REMOTE_ACCESS_TOKEN, 199
MSK_SPAR_SENSITIVITY_FILE_NAME, 199
MSK_SPAR_SENSITIVITY_RES_FILE_NAME, 199
MSK_SPAR_SOL_FILTER_XC_LOW, 199
MSK_SPAR_SOL_FILTER_XC_UPR, 200
MSK_SPAR_SOL_FILTER_XX_LOW, 200
MSK_SPAR_SOL_FILTER_XX_UPR, 200
MSK_SPAR_STAT_FILE_NAME, 200
MSK_SPAR_STAT_KEY, 200
MSK_SPAR_STAT_NAME, 200
MSK_SPAR_WRITE_LP_GEN_VAR_NAME, 200

Response codes

Termination, 201
"MSK_RES_OK", 201
"MSK_RES_TRM_INTERNAL", 201
"MSK_RES_TRM_INTERNAL_STOP", 201
"MSK_RES_TRM_MAX_ITERATIONS", 201
"MSK_RES_TRM_MAX_NUM_SETBACKS", 201
"MSK_RES_TRM_MAX_TIME", 201
"MSK_RES_TRM_MIO_NUM_BRANCHES", 201
"MSK_RES_TRM_MIO_NUM_RELAXS", 201
"MSK_RES_TRM_NUM_MAX_NUM_INT_SOLUTIONS", 201
"MSK_RES_TRM_NUMERICAL_PROBLEM", 201
"MSK_RES_TRM_OBJECTIVE_RANGE", 201
"MSK_RES_TRM_STALL", 201
"MSK_RES_TRM_USER_CALLBACK", 201
Warnings, 202
"MSK_RES_WRN_ANA_ALMOST_INT_BOUNDS", 203
"MSK_RES_WRN_ANA_C_ZERO", 203
"MSK_RES_WRN_ANA_CLOSE_BOUNDS", 203
"MSK_RES_WRN_ANA_EMPTY_COLS", 203
"MSK_RES_WRN_ANA_LARGE_BOUNDS", 203
"MSK_RES_WRN_DROPPED_NZ_QOBJ", 202
"MSK_RES_WRN_DUPLICATE_BARVARIABLE_NAMES", 203
"MSK_RES_WRN_DUPLICATE_CONE_NAMES", 203
"MSK_RES_WRN_DUPLICATE_CONSTRAINT_NAMES", 203
"MSK_RES_WRN_DUPLICATE_VARIABLE_NAMES", 203
"MSK_RES_WRN_ELIMINATOR_SPACE", 203
"MSK_RES_WRN_EMPTY_NAME", 203
"MSK_RES_WRN_EXP_CONES_WITH_VARIABLES_FIXED_AT_ZERO", 204
"MSK_RES_WRN_IGNORE_INTEGER", 202
"MSK_RES_WRN_INCOMPLETE_LINEAR_DEPENDENCY_CHECK", 203
"MSK_RES_WRN_LARGE_AIJ", 202
"MSK_RES_WRN_LARGE_BOUND", 202
"MSK_RES_WRN_LARGE_CJ", 202
"MSK_RES_WRN_LARGE_CON_FX", 202
"MSK_RES_WRN_LARGE_LO_BOUND", 202
"MSK_RES_WRN_LARGE_UP_BOUND", 202
"MSK_RES_WRN_LICENSE_EXPIRE", 202
"MSK_RES_WRN_LICENSE_FEATURE_EXPIRE", 203
"MSK_RES_WRN_LICENSE_SERVER", 203
"MSK_RES_WRN_LP_DROP_VARIABLE", 202
"MSK_RES_WRN_LP_OLD_QUAD_FORMAT", 202
"MSK_RES_WRN_MIO_INFEASIBLE_FINAL", 202
"MSK_RES_WRN_MPS_SPLIT_BOU_VECTOR", 202
"MSK_RES_WRN_MPS_SPLIT_RAN_VECTOR", 202
"MSK_RES_WRN_MPS_SPLIT_RHS_VECTOR", 202
"MSK_RES_WRN_NAME_MAX_LEN", 202
"MSK_RES_WRN_NO_DUALIZER", 204
"MSK_RES_WRN_NO_GLOBAL_OPTIMIZER", 202
"MSK_RES_WRN_NZ_IN_UPR_TRI", 202
"MSK_RES_WRN_OPEN_PARAM_FILE", 202
"MSK_RES_WRN_PARAM_IGNORED_CMIO", 203
"MSK_RES_WRN_PARAM_NAME_DOU", 203

"MSK_RES_WRN_PARAM_NAME_INT", 203
"MSK_RES_WRN_PARAM_NAME_STR", 203
"MSK_RES_WRN_PARAM_STR_VALUE", 203
"MSK_RES_WRN_POW_CONES_WITH_ROOT_FIXED_AT_ZERO", 204
"MSK_RES_WRN_PRESOLVE_OUTOFSPACE", 203
"MSK_RES_WRN_QUAD_CONES_WITH_ROOT_FIXED_AT_ZERO", 204
"MSK_RES_WRN_RQUAD_CONES_WITH_ROOT_FIXED_AT_ZERO", 204
"MSK_RES_WRN_SOL_FILE_IGNORED_CON", 202
"MSK_RES_WRN_SOL_FILE_IGNORED_VAR", 202
"MSK_RES_WRN_SOL_FILTER", 202
"MSK_RES_WRN_SPAR_MAX_LEN", 202
"MSK_RES_WRN_SYM_MAT_LARGE", 204
"MSK_RES_WRN_TOO_FEW_BASIS_VARS", 202
"MSK_RES_WRN_TOO_MANY_BASIS_VARS", 202
"MSK_RES_WRN_UNDEF_SOL_FILE_NAME", 202
"MSK_RES_WRN_USING_GENERIC_NAMES", 203
"MSK_RES_WRN_WRITE_CHANGED_NAMES", 203
"MSK_RES_WRN_WRITE_DISCARDED_CFIX", 203
"MSK_RES_WRN_ZERO_AIJ", 202
"MSK_RES_WRN_ZEROS_IN_SPARSE_COL", 203
"MSK_RES_WRN_ZEROS_IN_SPARSE_ROW", 203
Errors, 204
"MSK_RES_ERR_AD_INVALID_CODELIST", 215
"MSK_RES_ERR_API_ARRAY_TOO_SMALL", 214
"MSK_RES_ERR_API_CB_CONNECT", 214
"MSK_RES_ERR_API_FATAL_ERROR", 214
"MSK_RES_ERR_API_INTERNAL", 215
"MSK_RES_ERR_APPENDING_TOO_BIG_CONE", 212
"MSK_RES_ERR_ARG_IS_TOO_LARGE", 209
"MSK_RES_ERR_ARG_IS_TOO_SMALL", 209
"MSK_RES_ERR_ARGUMENT_DIMENSION", 208
"MSK_RES_ERR_ARGUMENT_IS_TOO_LARGE", 216
"MSK_RES_ERR_ARGUMENT_LENNEQ", 208
"MSK_RES_ERR_ARGUMENT_PERM_ARRAY", 211
"MSK_RES_ERR_ARGUMENT_TYPE", 208
"MSK_RES_ERR_BAR_VAR_DIM", 215
"MSK_RES_ERR_BASIS", 210
"MSK_RES_ERR_BASIS_FACTOR", 213
"MSK_RES_ERR_BASIS_SINGULAR", 213
"MSK_RES_ERR_BLANK_NAME", 206
"MSK_RES_ERR_CBF_DUPLICATE_ACOORD", 217
"MSK_RES_ERR_CBF_DUPLICATE_BCOORD", 217
"MSK_RES_ERR_CBF_DUPLICATE_CON", 217
"MSK_RES_ERR_CBF_DUPLICATE_INT", 217
"MSK_RES_ERR_CBF_DUPLICATE_OBJ", 217
"MSK_RES_ERR_CBF_DUPLICATE_OBJACOORD", 217
"MSK_RES_ERR_CBF_DUPLICATE_POW_CONES", 218
"MSK_RES_ERR_CBF_DUPLICATE_POW_STAR_CONES", 218
"MSK_RES_ERR_CBF_DUPLICATE_PSDVAR", 217
"MSK_RES_ERR_CBF_DUPLICATE_VAR", 217
"MSK_RES_ERR_CBF_INVALID_CON_TYPE", 217
"MSK_RES_ERR_CBF_INVALID_DIMENSION_OF_CONES", 218
"MSK_RES_ERR_CBF_INVALID_DOMAIN_DIMENSION", 217
"MSK_RES_ERR_CBF_INVALID_EXP_DIMENSION", 218
"MSK_RES_ERR_CBF_INVALID_INT_INDEX", 217
"MSK_RES_ERR_CBF_INVALID_NUMBER_OF_CONES", 218
"MSK_RES_ERR_CBF_INVALID_POWER", 218
"MSK_RES_ERR_CBF_INVALID_POWER_CONE_INDEX", 218
"MSK_RES_ERR_CBF_INVALID_POWER_STAR_CONE_INDEX", 218
"MSK_RES_ERR_CBF_INVALID_PSDVAR_DIMENSION", 217
"MSK_RES_ERR_CBF_INVALID_VAR_TYPE", 217
"MSK_RES_ERR_CBF_NO_VARIABLES", 217
"MSK_RES_ERR_CBF_NO_VERSION_SPECIFIED", 217
"MSK_RES_ERR_CBF_OBJ_SENSE", 217
"MSK_RES_ERR_CBF_PARSE", 217
"MSK_RES_ERR_CBF_POWER_CONE_IS_TOO_LONG", 218
"MSK_RES_ERR_CBF_POWER_CONE_MISMATCH", 218
"MSK_RES_ERR_CBF_POWER_STAR_CONE_MISMATCH", 218
"MSK_RES_ERR_CBF_SYNTAX", 217
"MSK_RES_ERR_CBF_TOO_FEW_CONSTRAINTS", 217
"MSK_RES_ERR_CBF_TOO_FEW_INTS", 217
"MSK_RES_ERR_CBF_TOO_FEW_PSDVAR", 217
"MSK_RES_ERR_CBF_TOO_FEW_VARIABLES", 217
"MSK_RES_ERR_CBF_TOO_MANY_CONSTRAINTS", 217
"MSK_RES_ERR_CBF_TOO_MANY_INTS", 217
"MSK_RES_ERR_CBF_TOO_MANY_VARIABLES", 217
"MSK_RES_ERR_CBF_UNHANDLED_POWER_CONE_TYPE", 218
"MSK_RES_ERR_CBF_UNHANDLED_POWER_STAR_CONE_TYPE", 218
"MSK_RES_ERR_CBF_UNSUPPORTED", 217
"MSK_RES_ERR_CON_Q_NOT_NSD", 211
"MSK_RES_ERR_CON_Q_NOT_PSD", 211
"MSK_RES_ERR_CONE_INDEX", 211
"MSK_RES_ERR_CONE_OVERLAP", 211
"MSK_RES_ERR_CONE_OVERLAP_APPEND", 212
"MSK_RES_ERR_CONE_PARAMETER", 212
"MSK_RES_ERR_CONE_REP_VAR", 211
"MSK_RES_ERR_CONE_SIZE", 211
"MSK_RES_ERR_CONE_TYPE", 212
"MSK_RES_ERR_CONE_TYPE_STR", 212
"MSK_RES_ERR_DATA_FILE_EXT", 205
"MSK_RES_ERR_DUP_NAME", 206
"MSK_RES_ERR_DUPLICATE_AIJ", 212
"MSK_RES_ERR_DUPLICATE_BARVARIABLE_NAMES", 216
"MSK_RES_ERR_DUPLICATE_CONE_NAMES", 216
"MSK_RES_ERR_DUPLICATE_CONSTRAINT_NAMES", 216
"MSK_RES_ERR_DUPLICATE_VARIABLE_NAMES", 216
"MSK_RES_ERR_END_OF_FILE", 206
"MSK_RES_ERR_FACTOR", 213

portfolio optimization, 68
cardinality constraints, 77
efficient frontier, 71
factor model, 72
market impact cost, 73
Markowitz model, 68
Pareto optimality, 68
slippage cost, 73
transaction cost, 75
positive semidefinite, 14
power cone, 19
power cone optimization, 19
presolve, 106
eliminator, 107
linear dependency check, 107
numerical issues, 107
primal
certificate, 96, 100
feasible, 95, 109, 115
infeasible, 95, 96, 100, 109, 116
problem, 96, 99, 102
solution, 38, 95
primal-dual
problem, 108, 115
solution, 96
problem
dual, 96, 99, 102
feasible, 95
infeasible, 95, 96, 102
load, 41
primal, 96, 99, 102
primal-dual, 108, 115
save, 40
status, 36
unbounded, 97, 101
PTF format, 284

Q

qo1
example, 14
quadratic
constraint, 103
quadratic cone, 17
quadratic objective
example, 14
quadratic optimization, 103
quality
solution, 120

R

relaxation, 119
repair
infeasibility, 123
response code, 39
rotated quadratic cone, 17

S

scaling, 107
semidefinite
cone, 23
infeasibility, 102
matrix, 23
variable, 23
semidefinite optimization, 23, 102
sensitivity analysis, 127
basis type, 129
shadow price, 129
simplex
linear optimization, 113
logging, 114
numerical issues, 113
optimizer, 113
parameter, 113
termination criteria, 113
slippage cost, 73
sol format, 296
solution
basic, 36
dual, 38
file format, 296
integer, 36
integer feasible, 120
interior-point, 36
near-optimal, 120
optimal, 37
primal, 38, 95
primal-dual, 96
quality, 120
retrieve, 36
status, 37
status
problem, 36
solution, 37
symmetric
matrix, 23

T

task format, 288
termination, 36
termination criteria, 43
conic optimization, 116
integer optimization, 120
interior-point, 110, 116
linear optimization, 110, 113
simplex, 113
tolerance, 111, 117, 120
thread, 66
time limit, 43
tolerance
integer optimization, 120
termination criteria, 111, 117, 120
transaction cost, 75
troubleshooting
installation, 5

U

unbounded
    problem, 97, 101
user callback, *see* callback

V

variable, 95, 98
    bound, 11, 95, 98
    dual, 96, 99
    integer, 29
    linear optimization, 11
    semidefinite, 23