The MOSEK optimization tools manual.
Version 5.0 (Revision 138).

www.mosek.com

Published by MOSEK ApS, Denmark.

Contact information

Phone	+45 3917 9907	
Fax	+45 3917 9823	
WEB	http://www.mosek.com	
Email	sales@mosek.com	Sales, pricing, and licensing.
	support@mosek.com	Technical support, questions and bug reports.
	info@mosek.com	Everything else.

Mail
MOSEK ApS
C/O Symbion Science Park
Fruebjergvej 3, Box 16
2100 Copenhagen Ø
Denmark

Contents

License agreement

Before using the MOSEK software, please read the license agreement available in the distribution incd

`mosek\5\license\index.html`

Chapter 1

Changes and new features in MOSEK

The section presents improvements and new features added to MOSEK in version 5.0.

1.1 File formats

- The `OSiL` XML format for linear problems is now supported as output-only format.
- The new Optimization Problem file Format (OPF) is now available. It incorporates linear, quadratic, and conic problems in a single format, as well as parameter settings and solutions.
- The `OBJNAME` section is now supported in the MPS format.

1.2 Optimizers

- The interior-point solver is about 20% faster on average for large linear problems, compared to MOSEK 4.0.
- The dual simplex solver is about 40% faster on average compared to MOSEK 4.0.
- For the primal simplex solver, handling of problems with long slim structure has been improved.
- For both simplex optimizers numerical stability, hot-start efficiency and degeneracy handling has been improved substantially.
- A simplex network flow optimizer is now available. In many cases the specialized simplex optimizer can solve a pure network flow optimization problem up to 10 times faster than the standard simplex optimizer.
- Presolve is now by default turned on for hot-start with the simplex optimizers.
- The mixed integer optimizer now includes the feasibility pump heuristic to find a good initial feasible solution.

- Full support for setting branching priorities on integer constrained variables.

1.3 License system

- The Flexlm license software has been upgraded to version 11.4.
- Dongles are supported in 64 bit Windows.

1.4 Other changes

- The documentation has been improved. Each interface now have a complete dedicated manual, and many code examples have been added. The HTML version has been subject to heavy cosmetical changes.

1.5 Interfaces

- A complete Python interface is now available.
- The MATLAB interface supports the MATLAB versions R2006a, R2006b, and R2007a.
- The general convex interface has been disabled in the Java and .NET interfaces.
- The Java API provides an interface to the native `scopt` functionality.

1.6 Supported platforms

- Mac OSX 32 bit for x86 version has been added.
- Solaris 32 bit for x86 version has been added
- Solaris 64 bit for x86 version has been added.

Chapter 2

The MOSEK optimization tools

2.1 What is MOSEK

MOSEK is a software package for solving mathematical optimization problems.

The core of MOSEK consists of a number of optimizers that can solve various optimization problems. The problem clases MOSEK is designed to solve are:

- Linear problems.
- Conic quadratic problems. (also known as second order optimization).
- General convex problems. In particular, MOSEK is wellsuited for:
  - Convex quadratic problems.
  - Convex quadratically constrained problems.
  - Geometric problems (posynomial case).
- Integer problems, i.e. problems where some of the variables are constrained to integer values.

These problem classes can be solved using an appropriate optimizer built into MOSEK:

- Interior-point optimizer for all continuous problems.
- Primal or dual simplex optimizer for linear problems.
- Conic interior-point optimizer for conic quadratic problems.
- Mixed-integer optimizer based on a branch and cut technology.

All the optimizers available in MOSEK are built for solving large-scale sparse problems and have been extensively tuned for stability and performance.

2.1.1 Interfaces

There are several ways to interface with MOSEK:

- Files:
    - MPS format: MOSEK reads the industry standard MPS file format for specifying (mixed integer) linear optimization problems. Moreover an MPS file can also be used to specify quadratic, quadratically constrained, and conic optimization problems.
    - LP format: MOSEK can read and write the CPLEX LP format with some restrictions.
    - OPF format: MOSEK also has its own text based format called OPF. The format is closely related to the LP but is much more robust in its specification
- APIs: MOSEK can also invoked from various programming languages.
    - C/C++,
    - C# (plus other .NET languages),
    - Delphi,
    - Java and
    - Python.
- Thrid party programs:
    - AMPL: MOSEK can easily be used from the modeling language AMPL[1] which is a high-level modeling language that makes it possible to formulate optimization problems in a language close to the original “pen and paper” model formulation.
    - MATLAB: When using the MOSEK optimization toolbox for Matlab the functionality of MOSEK can easily be used within MATLAB.

2.2 How to use this manual

This manual consists of two parts each consisting of several chapters.

The first part consists of the Chapters 4 to 14 and is a User’s guide which provides a quick introduction to the usage of MOSEK. The last part consists of appendixes A - I is a reference manual for the MOSEK command line tool, file formats and parameters.

---

[1] See http://www.ampl.com for further information.

Chapter 3

Getting support and help

3.1 MOSEK documentation

For an overview of the available MOSEK documentation please see

`mosek\5\help\index.html`

in the distribution.

3.2 Additional reading

In this manual it is assumed the reader is familiar with mathematics and in particular mathematical optimization. Some introduction to linear programming can be found in books such as “Linear programming” by Chvátal [13] or “Computer Solution of Linear Programs” by Nazareth [19]. For more theoretical aspects see for example “Nonlinear programming: Theory and algorithms” by Bazaraa, Shetty, and Sherali [11]. Finally the book “Model building in mathematical programming” by Williams [24] provides an excellent introduction to modelling issues in optimization.

Another useful resource is “Mathematical Programming Glossary” available at http://glossary.computing.society.informs.org

Chapter 4

Using the MOSEK command line tool

This chapter introduces the MOSEK command line tool which allows the user to solve optimization problems specified in a text file. The main reasons to use the command line tool are

- to solve small problems by hand, and
- as a debugging tool for large problems generated by other programs.

4.1 Getting started

The syntax for the mosek command line tool is

```
mosek [options] filename
```

`[options]` are some options which modify the behavior of MOSEK such as whether the optimization problem is minimized or maximized. `filename` is the name of the file which contains the problem data. E.g the

```
mosek -min afiro.mps
```

command line tells MOSEK to read data from the `afiro.mps` file and to minimize the objective function.

By default the solution to the optimization problem is stored in the files `afiro.sol` and `afiro.bas`. The `.sol` and `.bas` files contains the interior and basis solution respectively. For problems with integer variables the solution is written to a file with the extension `.int`.

For a complete list of command line parameters type

```
mosek -h
```

or see Appendix A.

4.2 Examples

Using several examples we will subsequently demonstrate how to use the MOSEK command line tool.

4.2.1 Linear optimization

A linear optimization problem is a problem where a linear objective function is optimized subject to linear constraints. An example of a linear optimization problem is

$$\begin{array}{lrcrcl} \text{minimize} & -10x_1 & & -9x_2, & & \\ \text{subject to} & 7/10x_1 & + & 1x_2 & \leq & 630, \\ & 1/2x_1 & + & 5/6x_2 & \leq & 600, \\ & 1x_1 & + & 2/3x_2 & \leq & 708, \\ & 1/10x_1 & + & 1/4x_2 & \leq & 135, \\ & x_1, & & x_2 & \geq & 0. \end{array} \tag{4.1}$$

The solution of the example (4.1) using MOSEK consists of three steps:

- Creating an input file describing the problem.
- Optimizing the problem using MOSEK.
- Viewing the solution reports.

The input file for MOSEK is a plain text file containing a description of the problem and it must be in either the MPS, the LP, or the OPF format. Below we present the example encoded as an OPF file:

```
[comment]
  Example lo1.mps converted to OPF.
[/comment]

[hints]
  # Give a hint about the size of the different elements in the problem.
  # These need only be estimates, but in this case they are exact.
  [hint NUMVAR] 2 [/hint]
  [hint NUMCON] 4 [/hint]
  [hint NUMANZ] 8 [/hint]
[/hints]

[variables]
  # All variables that will appear in the problem
  x1 x2
[/variables]

[objective minimize 'obj']
   - 10 x1 - 9 x2
[/objective]
```

```
[constraints]
  [con 'c1']  0.7 x1 +              x2 <= 630 [/con]
  [con 'c2']  0.5 x1 + 0.8333333333 x2 <= 600 [/con]
  [con 'c3']      x1 + 0.66666667   x2 <= 708 [/con]
  [con 'c4']  0.1 x1 + 0.25         x2 <= 135 [/con]
[/constraints]

[bounds]
  # By default all variables are free. The following line will
  # change this to all variables being nonnegative.
  [b] 0 <= * [/b]
[/bounds]
```

For details on the syntax of the OPF format please consult Appendix D.

After the input file has been created, the problem can be optimized. Assuming the input file has been given the name `lo1.opf`, then the problem is optimized using the command line

```
  mosek lo1.opf
```

Two solution report files `lo1.sol` and `lo1.bas` are generated where the first file contains the interior solution and the second file contains the basic solution. In this case the `lo1.bas` file has the format:

```
NAME                : EXAMPLE
PROBLEM STATUS      : PRIMAL_AND_DUAL_FEASIBLE
SOLUTION STATUS     : OPTIMAL
OBJECTIVE NAME      : obj
PRIMAL OBJECTIVE    : -7.66799999e+003
DUAL OBJECTIVE      : -7.66799999e+003

CONSTRAINTS
INDEX   NAME            AT ACTIVITY          LOWER LIMIT        UPPER LIMIT        DUAL LOWER         DUAL UPPER
1       c1              UL 6.30000000e+002   NONE               6.30000000e+002    0.00000000e+000    4.37499996e+000
2       c2              BS 4.80000000e+002   NONE               6.00000000e+002    0.00000000e+000    0.00000000e+000
3       c3              UL 7.08000000e+002   NONE               7.08000000e+002    0.00000000e+000    6.93750003e+000
4       c4              BS 1.17000000e+002   NONE               1.35000000e+002    0.00000000e+000    0.00000000e+000

VARIABLES
INDEX   NAME            AT ACTIVITY          LOWER LIMIT        UPPER LIMIT        DUAL LOWER         DUAL UPPER
1       x1              BS 5.39999998e+002   0.00000000e+000    NONE               0.00000000e+000    0.00000000e+000
2       x2              BS 2.52000001e+002   0.00000000e+000    NONE               0.00000000e+000    0.00000000e+000
```

The interpretation of the solution file should be obvious. E.g the optimal values of `x1` and `x2` are 539.99 and 252.00 respectively. A detailed discussion of the solution file format is given in Appendix F.

4.2.2 Quadratic optimization

An example of a quadratic optimization problem is

$$\begin{array}{lrcc} \text{minimize} & & & x_1^2 + 0.1x_2^2 + x_3^2 - x_1 x_3 - x_2 \\ \text{subject to} & 1 & \leq & x_1 + x_2 + x_3, \\ & & & x \geq 0. \end{array} \tag{4.2}$$

The problem is a quadratic optimization problem because all the constraints are linear and the objective can be stated on the form

$$0.5x^T Qx + c^T x$$

where in this particular case we have that

$$Q = \begin{bmatrix} 2 & 0 & -1 \\ 0 & 0.2 & 0 \\ -1 & 0 & 2 \end{bmatrix} \text{ and } c = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}. \tag{4.3}$$

MOSEK assumes that Q is symmetric and positive semi-definite. If these assumptions are not satisfied, MOSEK will most likely not compute a valid solution. Recall a matrix is symmetric if it satisfies the condition

$$Q = Q^T$$

and it is positive semi-definite if

$$x^T Q x \geq 0, \text{ for all } x.$$

An OPF file specifying the example can have the format:

```
[comment]
  Example qo1.mps conterted to OPF.
[/comment]

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

[variables]
  x1 x2 x3
[/variables]

[objective minimize 'obj']
   # The quadratic terms are often multiplied by 1/2,
   # but this is not required.

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

[constraints]
  [con 'c1'] 1 <= x1 + x2 + x3 [/con]
[/constraints]

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

Please note that the quadratic terms in objective are stated very naturally in the OPF format as follows

```
   - x2 + 0.5 ( 2 x1 ^ 2 - 2 x3 * x1 + 0.2 x2 ^ 2 + 2 x3 ^ 2 )
```

The example is solved using the

```
mosek qo1.opf
```

command line. In this case only one solution file named `qo1.sol` is produced. A `.bas` file is only produced for linear problems.

4.2.3 Conic optimization

Conic optimization is a generalization of linear optimization which allows the formulation of nonlinear convex optimization problems.

The main idea in conic optimization is to include constraints of the form

$$x^t \in \mathcal{C}$$

in the optimization problem where x^t consists of a subset of the variables and $\mathcal{C}$ is a convex cone. Recall that $\mathcal{C}$ is a convex cone if and only if $\mathcal{C}$ is a convex set and

$$x \in \mathcal{C} \Rightarrow \alpha x \in \mathcal{C} \text{ for all } \alpha \geq 0.$$

MOSEK cannot handle arbitrary conic constraints but only the two types

$$\left\{ x \in R^{n+1} : \; x_1 \geq \sqrt{\sum_{j=2}^{n+1} x_j^2} \right\} \tag{4.4}$$

and

$$\left\{ x \in R^{n+2} : \; 2x_1 x_2 \geq \sum_{j=2}^{n+2} x_j^2, \; x_1, x_2 \geq 0 \right\}. \tag{4.5}$$

(4.4) is called a quadratic cone whereas (4.5) is called a rotated quadratic cone.

Consider the problem

$$\begin{array}{lll} \text{minimize} & 1x_1 + 2x_2 & \\ \text{subject to} & \frac{1}{x_1} + \frac{2}{x_2} & \leq \quad 5, \\ & \;\; x \geq 0 & \end{array} \tag{4.6}$$

which may not initially look like a conic optimization problem. It can however be reformulated to

$$\begin{array}{llll} \text{minimize} & 1x_1 + 2x_2 & & \\ \text{subject to} & 2x_3 + 4x_4 & = & 5, \\ & \quad x_5^2 & \leq & 2x_1 x_3, \\ & \quad x_6^2 & \leq & 2x_2 x_4, \\ & \quad x_5 & = & 1, \\ & \quad x_6 & = & 1, \\ & x \geq 0. & & \end{array} \tag{4.7}$$

Problem (4.6) and problem (4.7) are equivalent because the constraints of (4.7)

$$\frac{x_5^2}{x_1} = \frac{1}{x_1} \leq 2x_3 \text{ and } \frac{x_6^2}{x_2} \leq \frac{1}{x_2} \leq 2x_4$$

and hence

$$\frac{1}{x_1} + \frac{2}{x_2} \leq 2x_3 + 4x_4 = 5.$$

The problem (4.7) is a conic quadratic optimization problem.

Using the MOSEK OPF format the problem can be represented as follows:

```
[comment]
  Example cqo1.mps conterted to OPF.
[/comment]

[hints]
  [hint NUMVAR] 6 [/hint]
  [hint NUMCON] 1 [/hint]
  [hint NUMANZ] 2 [/hint]
[/hints]

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

[objective minimize 'obj']
   x1 + 2 x2
[/objective]

[constraints]
  [con 'c1']  2 x3 + 4 x4 = 5 [/con]
[/constraints]

[bounds]
  # We let all variables default to the positive orthant
  [b] 0 <= * [/b]
  # ... and change those that differ from the default.
  [b] x5,x6 = 1 [/b]

  # We define two rotated quadratic cones

  # k1: 2 x1 * x3 >= x5^2
  [cone rquad 'k1'] x1, x3, x5 [/cone]

  # k2: 2 x2 * x4 >= x6^2
  [cone rquad 'k2'] x2, x4, x6 [/cone]
[/bounds]
```

For details on the OPF format please consult Appendix D. Finally, the example can be solved using the

```
  mosek cqo1.opf
```

command line call and the solution can be studied by inspecting the `cqo1.sol` file.

Format name	Standard	Read	Write	File type	File extension	Reference
OPF	No	Yes	Yes	ASCII/UTF8	opf	Appendix D
MPS	Yes	Yes	Yes	ASCII	mps	Appendix B
LP	Partially	Yes	Yes	ASCII	lp	Appendix C
OSiL XML	Yes	No	Yes	ASCII/UTF8	xml	Appendix E
Binary task format	No	Yes	Yes	Binary	mbt	

Table 4.1: Supported file formats.

4.3 Passing options to the command line tool

It is possible to modify the behavior of MOSEK by setting appropriate parameters. E.g assume that a linear optimization problem should be solved with the primal simplex optimizer rather than the default interior-point optimizer. This is done by setting the parameter `MSK_IPAR_OPTIMIZER` to the value `MSK_OPTIMIZER_PRIMAL_SIMPLEX`. To accomplish this append

```
-d MSK_IPAR_OPTIMIZER MSK_OPTIMIZER_PRIMAL_SIMPLEX.
```

to the command line. E.g the command

```
  mosek  -d MSK_IPAR_OPTIMIZER MSK_OPTIMIZER_PRIMAL_SIMPLEX lo1.opf
```

solves the problem specified by `lo1.opf` using the primal simplex optimizer. For further information on the parameters available in MOSEK please see Appendix H.

4.4 Reading and writing problem data files

MOSEK reads and writes problem data files in the formats presented in Table 4.1. The columns of Table 4.1 show:

- The name of the format.
- Whether the format is an industry standard format.
- If the format can be read by MOSEK.
- If the format can be written by MOSEK.
- The generic file type of the format, i.e. ASCII, UTF8, or binary.
- The file extension for the format
- The location of information about the format.

4.4.1 Reading compressed data files

MOSEK can read and write data files compressed with gzip [1]

For mosek to recognize a file as a gzip compressed file it must have the extension `.gz`. E.g the command

```
mosek myfile.mps.gz
```

will automatically decompress the file while reading it.

4.4.2 Converting from one format and to another

It is possible to use MOSEK to convert a problem file from one format to another. For instance assume the MPS file `myprob.mps` should be converted to an LP file named `myprob.lp`. This can be achieved with the command

```
mosek myprob.mps -out myprob.lp -x
```

Converting an MPS file to a LP file and back to an MPS file permutes the rows and columns of the original problem; this has no influence on the problem, but variables and constraints may appear in a different order.

4.5 Hot-start

Often a sequence of closely related optimization problems has to be solved. In such a case it can be expected that a previous optimal solution can serve as a good starting point when the modified problem is optimized.

Currently, only the simplex optimizer and the mixed-integer optimizer in MOSEK can exploit a guess for the optimal solution. The simplex optimizer can exploit an arbitrary guess for the optimal solution whereas the mixed-integer optimizer requires a feasible integer solution. For both the simplex optimizer and the mixed-integer optimizer it holds that if the guess is good then the optimizer may be able to reduce the solution time significantly when exploiting the guess.

4.5.1 An example

Assume that the example

$$\begin{array}{lccccc} \text{minimize} & c_1 x_1 & & -9x_2, & & \\ \text{subject to} & 7/10 x_1 & + & 1x_2 & \leq & 630, \\ & 1/2 x_1 & + & 5/6 x_2 & \leq & 600, \\ & 1x_1 & + & 2/3 x_2 & \leq & 708, \\ & 1/10 x_1 & + & 1/4 x_2 & \leq & 135, \\ & x_1, & & x_2 & \geq & 0. \end{array} \tag{4.8}$$

should be solved for c_1 identical to -5 and -10. Clearly, a solution for one c_1 value will also be feasible for another value. Therefore, it might be worthwhile to exploit the previous optimal solution when reoptimizing the problem.

---

[1] `gzip` is a public domain compression format. For further details about `gzip` consult http://www.gzip.org

Assume that two MPS files have been created each corresponding to one of the c_1 values then the commands[2]

```
mosek lo1.mps -baso .\lo1.bas
mosek lo1-b.mps -basi .\lo1.bas -baso .\lo1-b.bas
-d MSK_IPAR_OPTIMIZER MSK_OPTIMIZER_PRIMAL_SIMPLEX
```

demonstrates how to exploit the previous optimal solution in the second optimization.

In the first line MOSEK optimizes the first version of the optimization problem where c_1 is identical to -10. The `-baso .\lo1.bas` command line option makes sure that the optimal basic solution is written to the file `.\lo1.bas`.

In the second line the second instance of the problem is optimized. The `-basi .\lo1.bas` command line option forces MOSEK to read the previous optimal solution which MOSEK will try to exploit automatically. The `-baso .\lo1-b.bas` command line option makes sure that the optimal basic solution is written to the `.\lo1-b.bas` file. Finally, the

```
-d MSK_IPAR_OPTIMIZER MSK_OPTIMIZER_PRIMAL_SIMPLEX
```

command line option makes sure that the primal simplex optimizer is used for the reoptimization. This is important because the interior-point optimizer used by default does not exploit a previous optimal solution.

4.6 Further information

Additional information about the MOSEK command line tool is available in Appendix A.

4.7 Solution file filtering

The MOSEK solution files can be very space consuming for large problems. One way to cut down the solution file size is only to include variables which optimal value is in a certain interesting range i.e $[0.01, 0.99]$. This can be done by setting the MOSEK parameters

```
MSK_SPAR_SOL_FILTER_XX_LOW   0.01
MSK_SPAR_SOL_FILTER_XX_UPR   0.99
```

For further details consult the parameters `MSK_SPAR_SOL_FILTER_XC_LOW` and `MSK_SPAR_SOL_FILTER_XC_UPR`.

---

[2] The second line should not be broken into two separate lines.

Chapter 5

MOSEK and AMPL

AMPL is a modeling language for specifying linear and nonlinear optimization models in a natural way. AMPL also makes it easy to solve the problem and e.g. display the solution or part of it.

We will not discuss the specifics of the AMPL language here but instead refer the reader to [14] and the AMPL website http://www.ampl.com.

AMPL cannot solve optimization problems by itself but requires a link to an appropriate optimizer such as MOSEK. The MOSEK distribution includes an AMPL link which makes it possible to use MOSEK as an optimizer within AMPL.

5.1 Invoking the AMPL shell

The MOSEK distribution by default comes with the AMPL shell installed. To invoke the AMPL shell type:

```
mampl
```

5.2 Applicability

It is possible to specify problems in AMPL that cannot be solved by MOSEK. The optimization problem must be a smooth convex optimization problem as discussed in Section 9.5.

5.3 An example

In many instances, you can successfully apply MOSEK simply by specifying the model and data, setting the solver option to MOSEK, and typing solve. First to invoke the AMPL shell type:

```
mampl
```

when the AMPL shell has started type the commands:

```
ampl: model diet.mod;
ampl: data diet.dat;
```

Value	Message
0	the solution is optimal.
100	suboptimal primal solution.
101	superoptimal (dual feasible) solution.
150	the solution is near optimal.
200	primal infeasible problem.
300	dual infeasible problem.
400	too many iterations.
500	solution status is unknown.
501	ill-posed problem, solution status is unknown.
$\geq$ 501	The value - 501 is a MOSEK response code. See Appendix I.36 for all MOSEK response codes.

Table 5.1: Interpretation of `solve_result_num`.

```
ampl: option solver mosek;
ampl: solve;
```

The resulting output is:

```
MOSEK finished.
Problem status    - PRIMAL_AND_DUAL_FEASIBLE
Solution status   - OPTIMAL
Primal objective  - 14.8557377
Dual objective    - 14.8557377

Objective = Total_Cost
```

5.4 Determining the outcome of an optimization

The AMPL parameter `solve_result_num` is used to indicate the outcome of the optimization process. It is used as follows

```
ampl: display solve_result_num
```

Please refer to table 5.1 for possible values of this parameter.

5.5 Optimizer options

5.5.1 The MOSEK parameter database

The MOSEK optimizer has options and parameters controlling such things as the termination criterion and which optimizer is used. These parameters can be modified within AMPL as shown in the example below:

```
ampl: model diet.mod;
ampl: data diet.dat;
```

```
ampl: option solver mosek;
ampl: option mosek_options
ampl? 'msk_ipar_optimizer = msk_optimizer_primal_simplex \
ampl? msk_ipar_sim_max_iterations = 100000';
ampl: solve;
```

In the example above a string called `mosek_options` is created which contains the parameter settings. Each parameter setting has the format

```
parameter name = value
```

where "parameter name" can be any valid MOSEK parameter name. See Appendix H for a description of all valid MOSEK parameters.

An alternative way of specifying the options is

```
ampl: option mosek_options
ampl? 'msk_ipar_optimizer = msk_optimizer_primal_simplex'
ampl? ' msk_ipar_sim_max_iterations = 100000';
```

New options can also be appended to an existing option string as shown below

```
ampl: option mosek_options $mosek_options
ampl? ' msk_ipar_sim_print_freq = 0 msk_ipar_sim_max_iterations = 1000';
```

The expression `$mosek_options` expands to the current value of the option. Line two in the example appends an additional value `msk_ipar_sim_max_iterations` to the option string.

5.5.2 Options

5.5.2.1 `outlev`

MOSEK also recognizes the `outlev` option which controls the amount of printed output. 0 means no printed output and a higher value means more printed output. An example of setting `outlev` is as follows:

```
ampl: option mosek_options 'outlev=2';
```

5.5.2.2 `wantsol`

MOSEK recognize the option `wantsol`. We refer the reader to the AMPL manual [14] for details about this option.

5.6 Which solution is returned to AMPL

The MOSEK optimizer can produce three types of solutions: basic, integer, and interior point solutions. For nonlinear problems only an interior solution is available. For linear optimization problems optimized by the interior-point optimizer with basis identification turned on both a basic and an interior point solution are calculated. The simplex algorithm produces only a basic solution. Whenever both an interior and a basic solution are available, the basic solution is returned. For problems containing integer variables, the integer solution is returned to AMPL.

5.7 Hot-start

Frequently, a sequence of optimization problems is solved where each problem differs only slightly from the previous problem. In that case it may be advantageous to use the previous optimal solution to hot-start the optimizer. Such a facility is available in MOSEK only when the simplex optimizer is used.

The hot-start facility exploits the AMPL variable suffix `sstatus` to communicate the optimal basis back to AMPL, and AMPL uses this facility to communicate an initial basis to MOSEK. The following example demonstrates this feature.

```
ampl: model diet.mod;
ampl: model diet.dat;
ampl: option solver mosek;
ampl: option mosek_options
ampl? 'msk_ipar_optimizer = msk_optimizer_primal_simplex outlev=2';
ampl: solve;
ampl: display Buy.sstatus;
ampl: solve;
```

The resulting output is:

```
MOSEK 1.4: Accepted: msk_ipar_optimizer = MSK_OPTIMIZER_PRIMAL_SIMPLEX

ITER      DEGITER   FEAS          DOBJ
0         0         2.805e+003    0.0000000000e+000
6         0         0.000e+000    1.4855737705e+001
7         0         0.000e+000    1.4855737705e+001
Return code - 0 (MSK_RES_OK)
MOSEK finished.
Problem status    - PRIMAL_AND_DUAL_FEASIBLE
Solution status   - OPTIMAL
Primal objective  - 14.8557377
Dual objective    - 14.8557377

Objective = Total_Cost
Buy.sstatus [*] :=
'Quarter Pounder w/ Cheese'  bas
  'McLean Deluxe w/ Cheese'  low
                  'Big Mac'  low
               Filet-O-Fish  low
        'McGrilled Chicken'  low
             'Fries, small'  bas
         'Sausage McMuffin'  low
           '1% Lowfat Milk'  bas
             'Orange Juice'  low
;

MOSEK 1.4: Accepted: msk_ipar_optimizer = MSK_OPTIMIZER_PRIMAL_SIMPLEX

ITER      DEGITER   FEAS          DOBJ
```

```
0        0        0.000e+000    1.4855737705e+001
0        0        0.000e+000    1.4855737705e+001
1        0        0.000e+000    1.4855737705e+001
Return code - 0 (MSK_RES_OK)
MOSEK finished.
Problem status    - PRIMAL_AND_DUAL_FEASIBLE
Solution status   - OPTIMAL
Primal objective  - 14.8557377
Dual objective    - 14.8557377

Objective = Total_Cost
```

Please note that the second solve takes fewer iterations since the previous optimal basis is reused.

5.8 Sensitivity analysis

MOSEK can calculate sensitivity information for the objective and constraints. To enable sensitivity information set the option:

```
sensitivity = 1
```

Results are returned in variable/constraint suffixes as follows:

- `.down` Smallest value of objective coefficient/right hand side before the optimal basis changes.
- `.up` Largest value of objective coefficient/right hand side before the optimal basis changes.
- `.current` Current value of objective coefficient/right hand side.

For ranged constraints sensitivity information is returned only for the lower bound.

The example below returns sensitivity information on the `diet` model.

```
model diet.mod;
data diet.dat;
options solver mosek;
option mosek_options 'sensitivity=1';

solve;
#display sensitivity information and current solution.
display _var.down,_var.current,_var.up,_var;
#display sensitivity information and optimal dual values.
display _con.down,_con.current,_con.up,_con;
```

The resulting output is:

```
Return code - 0  [MSK_RES_OK]
MOSEK finished. (interior-point iterations - 11, primal simplex iterations - 0 dual simplex iteration
Problem status    : PRIMAL_AND_DUAL_FEASIBLE
Solution status   : OPTIMAL
Primal objective  : 88.2
```

```
Dual objective    : 88.2


suffix up OUT;
suffix down OUT;
suffix current OUT;
:  _var.down _var.current      _var.up        _var      :=
1   1.97182       3.19      Infinity          0
2   2.49818       2.59      Infinity          0
3   1.25636       2.29      Infinity          0
4   1.31455       2.89      Infinity          0
5   0.426429      1.89             1.94918   46.6667
6   1.89          1.99             2.44       0
7   1.90818       1.99      Infinity          0
8   1.35091       2.49      Infinity          0
;

:   _con.down _con.current   _con.up      _con       :=
1         700      700       3266.67   0.00181818
2   -Infinity      700        700      0
3         700      700        700      0.124182
4   -Infinity      700       1633.33   0
;
```

5.9 Using the command line version of the interface

AMPL can generate a data file which contains all the relevant problem information. Afterwards this data file can be loaded into MOSEK and the problem can be solved.

The following example demonstrate this feature.

```
mampl -obdiet prob.mod
mosek prob.nl outlev=1 -a
```

First AMPL is used to create the `prob.nl` data file and next MOSEK is used to solve the problem. Note that the `-a` command line option indicates that MOSEK is invoked in AMPL mode. When MOSEK is invoked in AMPL mode the normal MOSEK command line options should appear after the `-a` option except for the file name which should be the first argument. As the above example demonstrates MOSEK accepts command line options as specified by the AMPL “convention”. Which command line arguments MOSEK accepts in AMPL mode can be viewed by executing

```
mosek -= -a
```

To convert an AMPL file to one of the file formats MOSEK supports (e.g MPS) use the command below:

```
mosek prob.nl -a -out prob.mps -x
```

.

Chapter 6

MOSEK and GAMS

It is possible to call MOSEK from the GAMS modeling language . In order to do so a special GAMS/MOSEK link must be obtained from the GAMS Corporation.

Chapter 7

MOSEK and MATLAB

The MOSEK optimization toolbox for MATLAB is an easy to use interface to MOSEK that makes it possible to use MOSEK from within MATLAB.

The optimization toolbox is included in the MOSEK optimization tools distribution. See the separate documentation for the MATLAB toolbox for details.

Chapter 8

Intefaces to MOSEK

8.1 The optimizer API

The MOSEK optimizer API is an efficient interface to the optimizers implemented in MOSEK. E.g the interface makes it possible to call the linear optimizer from a C++ or Java program. The optimizer API is available for the languages

- C/C++/Delphi.
- Java.
- .NET (Visual Basic, C#, Managed C++, etc).
- Python.

Further details about the optimizer APIs are available at

`mosek\5\help\index.html`

or online at
http://www.mosek.com/documentation/

Chapter 9

Modelling

In this chapter we will discuss the following issues:

- The formal definitions of the problem types that MOSEK can solve.
- The solution information produced by MOSEK.
- The information produced by MOSEK if the problem is infeasible.
- A set of examples showing different ways of formulating commonly occurring problems so that they can be solved by MOSEK.
- Recommendations for formulating optimization problems.

9.1 Linear optimization

A linear optimization problem can be written as

$$\begin{array}{lrccl} \text{minimize} & & & c^T x + c^f & \\ \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \end{array} \tag{9.1}$$

where

- m is the number of constraints.
- n is the number of decision variables.
- $x \in R^n$ is a vector of decision variables.
- $c \in R^n$ is the linear part of the objective function.
- $A \in R^{m \times n}$ is the constraint matrix.
- $l^c \in R^m$ is the lower limit[1] on the activity for the constraints.

---

[1] We will use the words "bound" and "limit" interchangeably.

- $u^c \in R^m$ is the upper limit on the activity for the constraints.
- $l^x \in R^n$ is the lower limit on the activity for the variables.
- $u^x \in R^n$ is the upper limit on the activity for the variables.

A primal solution (x) is *(primal) feasible* if it satisfies all constraints in (9.1). If (9.1) has at least one primal feasible solution, then (9.1) is said to be (primal) feasible.

In case (9.1) does not have a feasible solution, the problem is said to be *(primal) infeasible*.

9.1.1 Duality for linear optimization

Corresponding to the primal problem (9.1), there is a dual problem

$$\begin{array}{lccc} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c & & \\ & +(l^x)^T s_l^x - (u^x)^T s_u^x + c^f & & \\ \text{subject to} & A^T y + s_l^x - s_u^x & = & c, \\ & -y + s_l^c - s_u^c & = & 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0. & & \end{array} \tag{9.2}$$

If a bound in the primal problem is plus or minus infinity, the corresponding dual variable is fixed at 0, and we use the convention that the product of the bound value and the corresponding dual variable is 0. For example

$$l_j^x = -\infty \;\Rightarrow\; (s_l^x)_j = 0 \text{ and } l_j^x \cdot (s_l^x)_j = 0.$$

This is equivalent to removing variable $(s_l^x)_j$ from the dual problem.

A solution

$$(y, s_l^c, s_u^c, s_l^x, s_u^x)$$

to the dual problem is feasible if it satisfies all the constraints in (9.2). If (9.2) has at least one feasible solution, then (9.2) is *(dual) feasible*, otherwise the problem is *(dual) infeasible*.

We will denote a solution

$$(x, y, s_l^c, s_u^c, s_l^x, s_u^x)$$

so that x is a solution to the primal problem (9.1), and

$$(y, s_l^c, s_u^c, s_l^x, s_u^x)$$

is a solution to the corresponding dual problem (9.2). A solution which is both primal and dual feasible is denoted a *primal-dual feasible primal-dual* solution.

9.1.1.1 A primal-dual feasible solution

Let

$$(x^*, y^*, (s_l^c)^*, (s_u^c)^*, (s_l^x)^*, (s_u^x)^*)$$

be a primal-dual feasible solution, and let

$$(x^c)^* := Ax^*.$$

For a primal-dual feasible solution we define the *optimality gap* as the difference between the primal and the dual objective value,

$$\begin{aligned} & c^T x^* + c^f - ((l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f) \\ = & \sum_{i=1}^{m} ((s_l^c)_i^* ((x_i^c)^* - l_i^c) + (s_u^c)_i^* (u_i^c - (x_i^c)^*) + \sum_{j=1}^{n} ((s_l^x)_j^* (x_j - l_j^x) + (s_u^x)_j^* (u_j^x - x_j^*)) \\ \geq & \; 0 \end{aligned}$$

where the first relation can be obtained by multiplying the dual constraints (9.2) by x and x^c respectively, and the second relation comes from the fact that each term in each sum is nonnegative. It follows that the primal objective will always be greater than or equal to the dual objective.

We then define the *duality gap* as the difference between the primal objective value and the dual objective value, i.e.

$$c^T x^* + c^f - ((l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f)$$

Please note that the duality gap will always be nonnegative.

9.1.1.2 An optimal solution

It is well-known that a linear optimization problem has an optimal solution if and only if there exist feasible primal and dual solutions such that the duality gap is zero, or, equivalently, that the *complementarity conditions*

$$\begin{array}{rcll} (s_l^c)_i^* ((x_i^c)^* - l_i^c) & = & 0, & i = 1, \ldots, m, \\ (s_u^c)_i^* (u_i^c - (x_i^c)^*) & = & 0, & i = 1, \ldots, m, \\ (s_l^x)_j^* (x_j - l_j^x) & = & 0, & j = 1, \ldots, n, \\ (s_u^x)_j^* (u_j^x - x_j^*) & = & 0, & j = 1, \ldots, n \end{array}$$

are satisfied.

If (9.1) has an optimal solution and MOSEK solves the problem successfully, both the primal and dual solution are reported, including a status indicating the exact state of the solution.

9.1.1.3 Primal infeasible problems

If the problem (9.1) is infeasible (has no feasible solution), MOSEK will report a primal certificate of the infeasibility: The dual solution reported is a certificate of infeasibility, and the primal solution is undefined.

A primal certificate (certificate of primal infeasibility) is a feasible solution to the modified dual problem

$$\begin{array}{ll} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x \\ \text{subject to} & \begin{array}{rcl} A^T y + s_l^x - s_u^x & = & 0, \\ -y + s_l^c - s_u^c & = & 0, \\ s_l^c, s_u^c, s_l^x, s_u^x \geq 0. & & \end{array} \end{array} \tag{9.3}$$

so that the objective is strictly positive, i.e. a solution

$$(y^*, (s_l^c)^*, (s_u^c)^*, (s_l^x)^*, (s_u^x)^*)$$

to (9.3) so that

$$(l^c)^T(s_l^c)^* - (u^c)^T(s_u^c)^* + (l^x)^T(s_l^x)^* - (u^x)^T(s_u^x)^* > 0.$$

Such a solution implies that (9.3) is unbounded, and that its dual is infeasible.

We note that the dual of (9.3) is a problem whose constraints are identical to the constraints of the original primal problem (9.1): If the dual of (9.3) is infeasible, so is the original primal problem.

9.1.1.4 Dual infeasible problems

If the problem (9.2) is infeasible (has no feasible solution), MOSEK will report a dual certificate of the infeasibility: The primal solution reported is a certificate of infeasibility, and the dual solution is undefined.

A certificate of dual infeasibility is a feasible solution to the problem

$$\begin{array}{lrclcl} \text{minimize} & & & c^T x & & \\ \text{subject to} & & & Ax - x^c & = & 0, \\ & \bar{l}^c & \leq & x^c & \leq & \bar{u}^c, \\ & \bar{l}^x & \leq & x & \leq & \bar{u}^x \end{array} \tag{9.4}$$

where

$$\bar{l}_i^c = \left\{ \begin{array}{ll} 0, & \text{if } l_i^c > -\infty, \\ -\infty & \text{otherwise} \end{array} \right. \quad \text{and} \quad \bar{u}_i^c := \left\{ \begin{array}{ll} 0, & \text{if } u_i^c < \infty, \\ \infty & \text{otherwise} \end{array} \right.$$

and

$$\bar{l}_j^x = \left\{ \begin{array}{ll} 0, & \text{if } l_j^x > -\infty, \\ -\infty & \text{otherwise} \end{array} \right. \quad \text{and} \quad \bar{u}_j^x := \left\{ \begin{array}{ll} 0, & \text{if } u_j^x < \infty, \\ \infty & \text{otherwise} \end{array} \right.$$

so that the objective value $c^T x$ is negative. Such a solution implies that (9.4) is unbounded, and that dual of (9.4) is infeasible.

We note that the dual of (9.4) is a problem whose constraints are identical to the constraints of the original dual problem (9.2): If the dual of (9.4) is infeasible, so is the original dual problem.

9.1.2 Primal and dual infeasible case

In case that both the primal problem (9.1) and the dual problem (9.2) are infeasible, MOSEK will report only one of the two possible certificates — which one is not defined (MOSEK returns the first certificate found).

9.2 Linear network flow problems

Network flow problems are a special class of linear optimization problems which has many applications. The class of network flow problems can be specified as follows. Let $G = (\mathcal{N}, \mathcal{A})$ be a directed network. Associated with every arc $(i,j) \in \mathcal{A}$ is a cost c_{ij} and a capacity $[l_{ij}^x, u_{ij}^x]$. Moreover, associated with each node $i \in \mathcal{N}$ in the network is a lower limit l_i^c and an upper limit u_i^c on the demand(supply) of the node. Now the minimum cost of a network flow problem can be stated as follows

$$\begin{array}{lrclcll} \text{minimize} & & & \sum\limits_{(i,j)\in\mathcal{A}} c_{ij} x_{ij} & & & \\ \text{subject to} & l_i^c & \leq & \sum\limits_{\{j:(i,j)\in\mathcal{A}\}} x_{ij} - \sum\limits_{\{j:(j,i)\in\mathcal{A}\}} x_{ji} & \leq & u_i^c & \forall i \in \mathcal{N}, \\ & l_{ij}^x & \leq & x_{ij} & \leq & u_{ij}^x & \forall (i,j) \in \mathcal{A}. \end{array} \tag{9.5}$$

A classical example of a network flow problem is the transportation problem, where the objective is to distribute goods from warehouses to customers at lowest possible total cost, see [2] for a detailed application reference.

It is well-known that problems with network flow structure can be solved efficiently with a specialized version of the simplex method. MOSEK includes a highly tuned network simplex implementation, see Section 10.3.1 for further details on how to invoke the network optimizer.

9.3 Quadratic and quadratically constrained optimization

A convex quadratic optimization problem is an optimization problem of the form

$$\begin{array}{lccccll} \text{minimize} & & & \frac{1}{2}x^TQ^ox + c^Tx + c^f & & & \\ \text{subject to} & l_k^c & \leq & \frac{1}{2}x^TQ^kx + \sum_{j=0}^{n-1} a_{k,i}x_j & \leq & u_k^c, & k = 0,\ldots,m-1, \\ & l^x & \leq & x & \leq & u^x, & j = 0,\ldots,n-1, \end{array} \tag{9.6}$$

where the convexity requirement implies that

- Q^o is a symmetric positive semi-definite matrix.
- If $l_k^c = -\infty$, then Q^k is a symmetric positive semi-definite matrix.
- If $u_k^c = \infty$, then Q^k is a symmetric negative semi-definite matrix.
- If $l_k > -\infty$ and $u_k^k < \infty$, then Q^k is a zero matrix.

The convexity requirement is very important and it is strongly recommended that MOSEK is applied to convex problems only.

9.3.1 A general recommendation

Any convex quadratic optimization problem can be reformulated as a conic optimization problem. It is our experience that for the majority of practical applications it is better to cast them as conic problems because

- the resulting problem is convex by construction, and
- the conic optimizer is more efficient than the optimizer for general quadratic problems.

See Section 9.4.4.1 for further details.

9.3.2 Reformulating as a separable quadratic problem

The simplest quadratic optimization problem is

$$\begin{array}{lccc} \text{minimize} & 1/2x^TQx + c^Tx & & \\ \text{subject to} & Ax & = & b, \\ & x \geq 0. & & \end{array} \tag{9.7}$$

The problem (9.7) is said to be a separable problem if Q is a diagonal matrix or, in other words, that the quadratic terms in the objective all have this form

$$x_j^2$$

instead of this form

$$x_j x_i.$$

The separable form has the following advantages:

- It is very easy to check the convexity assumption, and
- the simpler structure in a separable problem usually makes it easier to solve.

It is well-known that a positive semi-definite matrix Q can always be factorized, i.e. a matrix F exists so that

$$Q = F^T F. \tag{9.8}$$

In many practical applications of quadratic optimization F is known explicitly; for example if Q is a covariance matrix, F would be the set of observations producing it.

Using (9.8), the problem (9.7) can be reformulated as

$$\begin{array}{lccc} \text{minimize} & 1/2 y^T I y + c^T x & & \\ \text{subject to} & Ax & = & b, \\ & Fx - y & = & 0, \\ & x \geq 0. & & \end{array} \tag{9.9}$$

The problem (9.9) is also a quadratic optimization problem and has more constraints and variables than (9.7). However, the problem is separable. Normally, if F has fewer rows than columns, it is worthwhile to reformulate as a separable problem. Indeed consider the extreme case where F has one dense row and hence Q will be dense matrix.

The idea presented above is applicable to quadratic constraints too. Now consider the constraint

$$1/2 x^T (F^T F) x \leq b \tag{9.10}$$

where F is a matrix and b is a scalar. (9.10) can be reformulated as

$$\begin{array}{rcl} 1/2 y^T I y & \leq & b, \\ Fx - y & = & 0. \end{array}$$

It should be obvious how to generalize this idea to make any convex quadratic problem separable.

Next, consider the constraint

$$1/2 x^T (D + F^T F) x \leq b$$

where D is a positive semi-definite matrix, F is a matrix, and b is a scalar. We assume that D has a simple structure, i.e. D is for instance a diagonal or a block diagonal matrix. If this is the case, then it may be worthwhile performing the reformulation

$$\begin{array}{rcl} 1/2((x^T D x) + y^T I y) & \leq & b, \\ Fx - y & = & 0. \end{array}$$

Now the question may arise: When should a quadratic problem be reformulated to make it separable or near separable? The simplest rule of thumb is that it should be reformulated if the number of non-zeros used to represent the problem decreases when reformulating the problem.

9.4 Conic optimization

Conic optimization can be seen as a generalization of linear optimization. Indeed a conic optimization problem is a linear optimization problem plus a constraint of the form

$$x \in \mathcal{C}$$

where $\mathcal{C}$ is a convex cone. A complete conic problem has the form

$$\begin{array}{lccccl} \text{minimize} & & & c^T x + c^f & & \\ \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \\ & & & x \in \mathcal{C}. & & \end{array} \tag{9.11}$$

The cone $\mathcal{C}$ can be a Cartesian product of p convex cones, i.e.

$$\mathcal{C} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_p$$

in which case $x \in \mathcal{C}$ can be written as

$$x = (x_1, \ldots, x_p), \ x_1 \in \mathcal{C}_1, \ldots, x_p \in \mathcal{C}_p$$

where each $x_t \in R^{n_t}$. Please note that the n-dimensional Euclidean space R^n is a cone itself, so simple linear variables are still allowed.

MOSEK supports only a limited number of cones, specifically

$$\mathcal{C} = \mathcal{C}_1 \times \cdot \times \mathcal{C}_p$$

where each $\mathcal{C}_t$ has one of the following forms

- R set:
$$\mathcal{C}_t = \{x \in R^{n^t}\}.$$
- Quadratic cone:
$$\mathcal{C}_t = \left\{ x \in R^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\}.$$
- Rotated quadratic cone:
$$\mathcal{C}_t = \left\{ x \in R^{n^t} : 2x_1 x_2 \geq \sum_{j=3}^{n^t} x_j^2, \ x_1, x_2 \geq 0 \right\}.$$

Although these cones may seem to provide only limited expressive power they can be used to model a large range of problems as demonstrated in Section 9.4.4.

9.4.1 Duality for conic optimization

The dual problem corresponding to the conic optimization problem (9.11) is given by

$$\begin{array}{lccc}
\text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c & & \\
& +(l^x)^T s_l^x - (u^x)^T s_u^x + c^f & & \\
\text{subject to} & A^T y + s_l^x - s_u^x + s_n^x & = & c, \\
& -y + s_l^c - s_u^c & = & 0, \\
& s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0, \\
& s_n^x \in \mathcal{C}^* & &
\end{array} \tag{9.12}$$

where the dual cone $\mathcal{C}^*$ is a product of cones

$$\mathcal{C}^* = \mathcal{C}_1^* \times \cdots \mathcal{C}_p^*$$

where each $\mathcal{C}_t^*$ is the dual cone of $\mathcal{C}_t$. For the cone types MOSEK can handle, the relation between the primal and dual cone is given as follows:

- R set:
$$\mathcal{C}_t = \left\{ x \in R^{n^t} \right\} \quad \Leftrightarrow \quad \mathcal{C}_t^* := \left\{ s \in R^{n^t} \ : \ s = 0 \right\}.$$

- Quadratic cone:
$$\mathcal{C}_t := \left\{ x \in R^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\} \quad \Leftrightarrow \quad \mathcal{C}_t^* = \mathcal{C}_t.$$

- Rotated quadratic cone:
$$\mathcal{C}_t := \left\{ x \in R^{n^t} : 2x_1 x_2 \geq \sum_{j=3}^{n^t} x_j^2, \ x_1, x_2 \geq 0 \right\}. \quad \Leftrightarrow \quad \mathcal{C}_t^* = \mathcal{C}_t.$$

9.4.2 The dual of the dual

The dual problem corresponding to the dual problem is the primal problem.

9.4.3 Infeasibility

In case MOSEK finds a problem to be infeasible it will report a certificate of the infeasibility. This works exactly as for linear problems (see sections 9.1.1.3 and 9.1.1.4).

9.4.4 Examples

This section contains several examples of inequalities and problems that can be cast as conic optimization problems.

9.4.4.1 Quadratic objective and constraints

From Section 9.3.2 we know that any convex quadratic problem can be stated on the form

$$\begin{array}{ll} \text{minimize} & 0.5\,\|Fx\|^2 + c^T x, \\ \text{subject to} & 0.5\,\|Gx\|^2 + a^T x \quad \leq \quad b, \end{array} \tag{9.13}$$

where F and G are matrices and c and a are vectors. For simplicity we assume that there is only one constraint, but it should be obvious how to generalize the methods to an arbitrary number of constraints.

Problem (9.13) can be reformulated as

$$\begin{array}{lrcl} \text{minimize} & 0.5\,\|t\|^2 + c^T x, & & \\ \text{subject to} & 0.5\,\|z\|^2 + a^T x & \leq & b, \\ & Fx - t & = & 0, \\ & Gx - z & = & 0 \end{array} \tag{9.14}$$

after the introduction of the new variables t and z. It is easy to convert this problem to a conic quadratic optimization problem, i.e.

$$\begin{array}{lrcll} \text{minimize} & v + c^T x, & & & \\ \text{subject to} & p + a^T x & = & b, & \\ & Fx - t & = & 0, & \\ & Gx - z & = & 0, & \\ & w & = & 1, & \\ & q & = & 1, & \\ & \|t\|^2 & \leq & 2vw, & v, w \geq 0, \\ & \|z\|^2 & \leq & 2pq, & p, q \geq 0. \end{array} \tag{9.15}$$

In this case we can model the last two inequalities using rotated quadratic cones.

If we assume that F is a non-singular matrix — for instance a diagonal matrix — then

$$x = F^{-1} t.$$

and hence we can eliminate x from the problem to obtain:

$$\begin{array}{lrcll} \text{minimize} & v + c^T F^{-1} t, & & & \\ \text{subject to} & p + a^T F^{-1} t & = & b, & \\ & V F^{-1} t - z & = & 0, & \\ & w & = & 1, & \\ & q & = & 1, & \\ & \|t\|^2 & \leq & 2vw, & v, w \geq 0, \\ & \|z\|^2 & \leq & 2pq, & p, q \geq 0. \end{array} \tag{9.16}$$

In most cases MOSEK will perform this reduction automatically during the presolve phase before the optimization is performed.

9.4.4.2 Minimizing a sum of norms

The next example is the problem of minimizing a sum of norms i.e. the problem

$$\begin{array}{lccl} \text{minimize} & \sum_{i=1}^{k} \left\|x^i\right\| & & \\ \text{subject to} & Ax & = & b, \end{array} \tag{9.17}$$

where

$$x := \begin{bmatrix} x^1 \\ \vdots \\ x^k \end{bmatrix}.$$

This problem is equivalent to

$$\begin{array}{lccll} \text{minimize} & \sum_{i=1}^{k} z_i & & & \\ \text{subject to} & Ax & = & b, & \\ & \left\|x^i\right\| & \leq & z_i, & i = 1, \ldots, k, \end{array} \tag{9.18}$$

which in turn is equivalent to

$$\begin{array}{lccl} \text{minimize} & \sum_{i=1}^{k} z_i & & \\ \text{subject to} & Ax & = & b, \\ & (z_i, x^i) \in \mathcal{C}_i, & & i = 1, \ldots, k \end{array} \tag{9.19}$$

where all $\mathcal{C}^i$ are of the quadratic type, i.e.

$$\mathcal{C}_i := \left\{(z_i, x^i) : \ z_i \geq \left\|x^i\right\|\right\}.$$

The dual problem corresponding to (9.19) is

$$\begin{array}{lccll} \text{maximize} & b^T y & & & \\ \text{subject to} & A^T y + s & = & c, & \\ & t_i & = & 1, & i = 1, \ldots, k, \\ & (t_i, s^i) \in \mathcal{C}_i, & & & i = 1, \ldots, k \end{array} \tag{9.20}$$

where

$$s := \begin{bmatrix} s^1 \\ \vdots \\ s^k \end{bmatrix}.$$

This problem is equivalent to

$$\begin{array}{lccll} \text{maximize} & b^T y & & & \\ \text{subject to} & A^T y + s & = & c, & \\ & \left\|s^i\right\|_2^2 & \leq & 1, & i = 1, \ldots, k. \end{array} \tag{9.21}$$

Please note that the dual problem can be reduced to an "ordinary" convex quadratically constrained optimization problem in this case due to the special structure of the primal problem. In some cases it turns out that it is much better to solve the dual problem (9.20) rather than the primal problem (9.19).

9.4.4.3 Modelling polynomial terms using conic optimization

Generally an arbitrary polynomial term of the form

$$fx^g$$

cannot be represented with conic quadratic constraints, however in the following we will demonstrate some special cases where it is possible.

A particular simple polynomial term is the reciprocal, i.e.

$$\frac{1}{x}.$$

Now, a constraint of the form

$$\frac{1}{x} \leq y$$

where it is required that $x > 0$ is equivalent to

$$1 \leq xy \text{ and } x > 0$$

which in turn is equivalent to

$$\begin{array}{rcl} z & = & \sqrt{2}, \\ z^2 & \leq & 2xy. \end{array}$$

The last formulation is a conic constraint plus a simple linear equality.

For example, consider the problem

$$\begin{array}{lccc} \text{minimize} & c^T x & & \\ \text{subject to} & \sum_{j=1}^{n} \frac{f_j}{x_j} & \leq & b, \\ & x \geq 0, & & \end{array}$$

where it is assumed that $f_j > 0$ and $b > 0$. This problem is equivalent to

$$\begin{array}{lcccl} \text{minimize} & c^T x & & & \\ \text{subject to} & \sum_{j=1}^{n} z_j & = & b, & \\ & v_j & = & \sqrt{2}, & j = 1, \ldots, n, \\ & v_j^2 & \leq & 2z_j x_j, & j = 1, \ldots, n, \\ & x, z \geq 0, & & & \end{array} \tag{9.22}$$

because

$$v_j^2 = 2 \leq 2z_j x_j$$

implies that

$$\frac{1}{x_j} \leq z_j \text{ and } \sum_{j=1}^{n} \frac{f_j}{x_j} \leq \sum_{j=1}^{n} f_j z_j = b.$$

The problem (9.22) is a conic quadratic optimization problem having n 3 dimensional rotated quadratic cones.

The next example is the constraint

$$\begin{array}{rcl} \sqrt{x} & \geq & |t|, \\ x & \geq & 0, \end{array}$$

where both t and x are variables. This set is identical to the set

$$\begin{array}{rcl} t^2 & \leq & 2xz, \\ z & = & 0.5, \\ x, z, & \geq & 0. \end{array} \tag{9.23}$$

Occasionally when modelling the *market impact* term in portfolio optimization, the polynomial term $x^{\frac{3}{2}}$ occurs. Therefore, consider the set defined by the inequalities

$$\begin{array}{rcl} x^{1.5} & \leq & t, \\ 0 & \leq & x. \end{array} \tag{9.24}$$

We will exploit that $x^{1.5} = x^2/\sqrt{x}$. First define the set

$$\begin{array}{rcl} x^2 & \leq & 2st, \\ s, t & \geq & 0. \end{array} \tag{9.25}$$

Now, if we can make sure that

$$2s \leq \sqrt{x},$$

then we have the desired result since this implies that

$$x^{1.5} = \frac{x^2}{\sqrt{x}} \leq \frac{x^2}{2s} \leq t.$$

Please note that s can be chosen freely and that $\sqrt{x} = 2s$ is a valid choice.

Let

$$\begin{array}{rcl} x^2 & \leq & 2st, \\ w^2 & \leq & 2vr, \\ x & = & v, \\ s & = & w, \\ r & = & \frac{1}{8}, \\ s, t, v, r & \geq & 0, \end{array} \tag{9.26}$$

then

$$\begin{array}{rcl} s^2 & = & w^2 \\ & \leq & 2vr \\ & \leq & \frac{v}{4} \\ & = & \frac{x}{4}. \end{array}$$

Moreover,

$$\begin{array}{rcl} x^2 & \leq & 2st, \\ & \leq & 2\sqrt{\frac{x}{4}}t \end{array}$$

leading to the conclusion

$$x^{1.5} \leq t.$$

(9.26) is a conic reformulation which is equivalent to (9.24). Please note that the $x \geq 0$ constraint does not appear explicitly in (9.25) and (9.26), but implicitly since $x = v \geq 0$.

Finally, it should be mentioned that any polynomial term of form x^g where g is a positive rational number can be represented using conic quadratic constraints [3, pp. 12-13]

9.4.4.4 Further reading

If you want to know more about what can be modelled as a conic optimization problem we recommend the references [17, 12, 3].

9.4.5 Potential pitfalls in conic optimization

While a linear optimization problem either has a bounded optimal solution or is infeasible, the conic case is not as simple as that.

9.4.5.1 Non-attainment in the primal problem

Consider the example

$$\begin{array}{lrcl} \text{minimize} & z & & \\ \text{subject to} & 2yz & \geq & x^2, \\ & x & = & \sqrt{2}, \\ & y, z & \geq & 0. \end{array} \tag{9.27}$$

which corresponds to the problem

$$\begin{array}{lrcl} \text{minimize} & \frac{1}{y} & & \\ \text{subject to} & y & \geq & 0. \end{array} \tag{9.28}$$

Clearly, the optimal objective value is zero but it is never attained because implicitly we assume that the optimal y should be finite.

9.4.5.2 Non-attainment in the dual problem

Next, consider the example

$$\begin{array}{lrcl} \text{minimize} & x_4 & & \\ \text{subject to} & x_3 + x_4 & = & 1, \\ & x_1 & = & 0, \\ & x_2 & = & 1, \\ & 2x_1x_2 & \geq & x_3^2, \\ & x_1x_2 & \geq & 0. \end{array} \tag{9.29}$$

which has the optimal solution

$$x_1^* = 0,\ x_2^* = 1,\ x_3^* = 0 \text{ and } x_4^* = 1$$

implying that the optimal primal objective value is 1.

Now, the dual problem corresponding to (9.29) is

$$\begin{array}{lrcl} \text{maximize} & y_1 + y_3 & & \\ \text{subject to} & y_2 + s_1 & = & 0, \\ & y_3 + s_2 & = & 0, \\ & y_1 + s_3 & = & 0, \\ & y_1 & = & 1, \\ & 2s_1s_2 & \geq & s_3^2, \\ & s_1s_2 & \geq & 0. \end{array} \tag{9.30}$$

Therefore,

$$y_1^* = 1$$

and

$$s_3^* = -1.$$

This implies that

$$2 s_1^* s_2^* \geq (s_3^*)^2 = 1$$

and hence $s_2^* > 0$. Given this fact we can conclude that

$$\begin{array}{rcl} y_1^* + y_3^* & = & 1 - s_2^* \\ & < & 1 \end{array}$$

implying that the optimal dual objective value is 1, however this is never attained. Hence, there no primal and dual bounded optimal solution that has zero duality gap exists. Of course it is possible to find a primal and dual feasible solution such that the duality gap is close to zero, however, s_1^* will be very large (unless a large duality gap is allowed). This is likely to make the problem (9.29) hard to solve.

An inspection of problem (9.29) reveals the constraint $x_1 = 0$, which implies that $x_3 = 0$. If we either add the redundant constraint

$$x_3 = 0$$

to the problem (9.29) or eliminate x_1 and x_3 from the problem it becomes easy to solve.

9.5 Nonlinear convex optimization

MOSEK is capable of solving smooth convex nonlinear optimization problems of the form

$$\begin{array}{lrcccl} \text{minimize} & & & f(x) + c^T x & & \\ \text{subject to} & & & g(x) + Ax - x^c & = & 0, \\ & l^c & \leq & x^c & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \end{array} \tag{9.31}$$

where

- m is the number of constraints.
- n is the number of decision variables.
- $x \in R^n$ is a vector of decision variables.
- $x^c \in R^m$ is a vector of constraints or slack variables.
- $c \in R^n$ is the linear part objective function.
- $A \in R^{m \times n}$ is the constraint matrix.
- $l^c \in R^m$ is the lower limit[2] on the activity for the constraints.

---

[2] We will use the words "bound" and "limit" interchangeably.

- $u^c \in R^m$ is the upper limit on the activity for the constraints.
- $l^x \in R^n$ is the lower limit on the activity for the variables.
- $u^x \in R^n$ is the upper limit on the activity for the variables.
- $f : R^n \rightarrow R$ is a nonlinear function.
- $g : R^n \rightarrow R^m$ is a nonlinear vector function.

This means that the ith constraint has the form

$$l_i^c \leq g_i(x) + \sum_{j=1}^{n} a_{i,j} x_j \leq u_i^c$$

when the x_i^c variable has been eliminated.

The linear term Ax is not included in $g(x)$ since it can be handled much more efficiently as a separate entity when optimizing.

The nonlinear functions f and g must be smooth (twice differentiable) in all $x \in [l^x; u^x]$. Moreover, $f(x)$ must be a convex function and $g_i(x)$ must satisfy

$$\begin{array}{rcl} l_i^c = -\infty & \Rightarrow & g_i(x) \quad \text{is convex,} \\ u_i^c = \infty & \Rightarrow & g_i(x) \quad \text{is concave,} \\ -\infty < l_i^c \leq u_i^c < \infty & \Rightarrow & g_i(x) = 0. \end{array}$$

9.5.1 Duality

So far, we have not discussed what happens when MOSEK is used to solve a primal or dual infeasible problem. In the subsequent section these issues are addressed.

Similar to the linear case, MOSEK reports dual information in the general nonlinear case. Indeed in this case the Lagrange function is defined by

$$\begin{array}{rcl} L(x^c, x, y, s_l^c, s_u^c, s_l^x, s_u^x) & := & f(x) + c^T x + c^f \\ & & -y^T (Ax + g(x) - x^c) \\ & & -(s_l^c)^T (x^c - l^c) - (s_u^c)^T (u^c - x^c) \\ & & -(s_l^x)^T (x - l^x) - (s_u^x)^T (u^x - x). \end{array}$$

and the dual problem is given by

$$\begin{array}{lrcl} \text{maximize} & L(x^c, x, y, s_l^c, s_u^c, s_l^x, s_u^x) & & \\ \text{subject to} & \nabla_{(x^c,x)} L(x^c, x, y, s_l^c, s_u^c, s_l^x, s_u^x) & = & 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0. & & \end{array}$$

which is equivalent to

$$\begin{array}{lrcl} \text{maximize} & f(x) - y^T g(x) - x^T (\nabla f(x)^T - \nabla g(x)^T y) & & \\ & +((l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f & & \\ \text{subject to} & -\nabla f(x)^T + A^T y + \nabla g(x)^T y + s_l^x - s_u^x & = & c, \\ & -y + s_l^c - s_u^c & = & 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0. & & \end{array} \tag{9.32}$$

9.6 Recommendations

Often an optimization problem can be formulated in several different ways, and the exact formulation used may have a significant impact on the solution time and the quality of the solution. In some cases the difference between a “good” and a “bad” formulation means the ability to solve the problem or not.

Below is a list of several issues that you should be aware of when developing a good formulation.

1. Sparsity is very important. The constraint matrix A is assumed to be a sparse matrix, where sparse means that it contains many zeros (typically less than 10% non-zeros). Normally, when A is sparser, less memory is required to store the problem and it can be solved faster.

2. Avoid large bounds as these can introduce all sorts of numerical problems. Assume that a variable x_j has the bounds

$$0.0 \leq x_j \leq 1.0e16.$$

   The number 1.0e16 is large and it is very likely that the constraint $x_j \leq 1.0e16$ is non-binding at optimum, and therefore that the bound $1.0e16$ will not cause problems. Unfortunately, this is a naïve assumption because the bound $1.0e16$ may actually affect the presolve, the scaling, the computation of the dual objective value, etc. In this case the constraint $x_j \geq 0$ is likely to be sufficient, i.e. $1.0e16$ is just a way of representing infinity.

3. Avoid large penalty terms in the objective, i.e. do not have large terms in the linear part of the objective function. They will most likely cause numerical problems.

4. On a computer all computations are performed in finite precision, which implies that

$$1 = 1 + \varepsilon$$

   where ε is about 10^{-16}. This means that the results of all computations are truncated leading to the introduction of rounding errors. The upshot is that very small numbers and very large numbers should be avoided, e.g. it is recommended that all elements in A are either zero or belong to the interval $[10^{-6}, 10^{6}]$. The same holds for the bounds and the linear objective.

5. Decreasing the number of variables or constraints does not *necessarily* make it easier to solve a problem. In certain cases, i.e. in nonlinear optimization, it might be a good idea to introduce more constraints and variables if it makes the model separable. Also a big but sparse problem might be advantageous compared to a smaller but denser problem.

6. Try to avoid linearly dependent rows among the linear constraints. Network flow problems and multi-commodity network flow problems, for example, often contain one or more linearly dependent rows.

7. Finally, it is recommended to consult some of the papers about preprocessing to get some ideas about efficient formulations. See e.g. [4, 5, 15, 16].

9.6.1 Avoid nearly infeasible models

Consider the linear optimization problem

$$\begin{array}{lrcl} \text{minimize} & & & \\ \text{subject to} & x+y & \leq & 10^{-10}+\alpha, \\ & 1.0e4x+2.0e4y & \geq & 10^{-6}, \\ & x,y\geq 0. & & \end{array} \tag{9.33}$$

Clearly, the problem is feasible for $\alpha = 0$. However, for $\alpha = -1.0e-10$ the problem is infeasible. This implies that an insignificant change in the right side of the constraints makes the problem status switch from feasible to infeasible. Such a model should be avoided.

9.7 Examples continued

9.7.1 The absolute value

Assume we have a constraint for the form

$$|f^T x + g| \leq b \tag{9.34}$$

where $x \in R^n$ is a vector of variables, and $f \in R^n$ and $g, b \in R$ are constants.

It is easy to verify that the constraint (9.34) is equivalent to

$$-b \leq f^T x + g - t \leq b \tag{9.35}$$

which is a set of ordinary linear inequality constraints.

Please note that equalities involving and absolute value such as

$$|x| = 1$$

cannot be formulated as a linear or even a convex optimization problem. It requires integer optimization.

9.7.2 The Markowitz portfolio model

In this section we will show how to model several versions of the Markowitz portfolio model using conic optimization.

The Markowitz portfolio model deals with the problem of selecting a portfolio of assets i.e. stocks, bonds, etc. The goal is to find a portfolio such that for a given return the risk is minimized. The assumptions are:

- A portfolio can consist of n traded assets numbered $1, 2, \ldots$ held over a period of time.
- w_j^0 is the initial holding of asset j where $\sum_j w_j^0 > 0$.
- r_j is the return on asset j and is assumed to be a random variable. r has known mean $\bar{r}$ and covariance Σ.

The variable x_j denotes the amount of asset j traded in the given period of time and has the following meaning:

- If $x_j > 0$, then the amount of asset j is increased (by purchasing).
- If $x_j < 0$, then the amount of asset j is decreased (by selling).

The model deals with two central quantities:

- Expected return:
$$E[r^T(w^0+x)] = \bar{r}^T(w^0+x).$$
- Variance (Risk):
$$V[r^T(w^0+x)] = (w^0+x)^T\Sigma(w^0+x).$$

By definition Σ is positive semi-definite and

$$\begin{array}{rcl} \text{Std. dev.} & = & \left\|\Sigma^{\frac{1}{2}}(w^0+x)\right\| \\ & = & \left\|L^T(w^0+x)\right\| \end{array}$$

where L is **any** matrix such that
$$\Sigma = LL^T$$
A low rank of Σ is advantageous from a computational point of view. A valid L can always be computed as the Cholesky factorization of Σ.

9.7.2.1 Minimizing variance for a given return

In our first model we want to minimize the variance while selecting a portfolio with a specified expected target return t. Additionally the portfolio must satisfy the budget (self-financing) constraint asserting that the total amount of assets sold must equal the total amount of assets purchased. This is expressed in the model

$$\begin{array}{lrcl} \text{minimize} & V[r^T(w^0+x)] & & \\ \text{subject to} & E[r^T(w^0+x)] & = & t, \\ & e^Tx & = & 0, \end{array} \tag{9.36}$$

where $e := (1,\ldots,1)^T$. Using the definitions above this may be formulated as a quadratic optimization problem:

$$\begin{array}{lrcl} \text{minimize} & (w^0+x)^T\Sigma(w^0+x) & & \\ \text{subject to} & \bar{r}^T(w^0+x) & = & t, \\ & e^Tx & = & 0, \end{array} \tag{9.37}$$

9.7.2.2 Conic quadratic reformulation.

An equivalent conic quadratic reformulation is given by:

$$\begin{array}{lrcl} \text{minimize} & f & & \\ \text{subject to} & \Sigma^{\frac{1}{2}}(w^0+x) - g & = & 0, \\ & \bar{r}^T(w^0+x) & = & t, \\ & e^Tx & = & 0, \\ & f \geq \|g\|. & & \end{array} \tag{9.38}$$

Here we minimize the standard deviation instead of the variance. Please note that $\Sigma^{\frac{1}{2}}$ can be replaced by any matrix L where $\Sigma = LL^T$. A low rank L is computationally advantageous.

9.7.2.3 Transaction costs with market impact term

We will now expand our model to include transaction costs as a fraction of the traded volume. [1, pp. 445-475] argues that transaction costs can be modelled as follows

$$\text{commission} + \frac{\text{bid}}{\text{ask}} - \text{spread} + \theta\sqrt{\frac{\text{trade volume}}{\text{daily volume}}}, \tag{9.39}$$

and that these are important to incorporate into the model.

In the following we deal with the last of these terms denoted the *market impact term*. If you sell (buy) a lot of assets the price is likely to go down (up). This can be captured in the market impact term

$$\theta\sqrt{\frac{\text{trade volume}}{\text{daily volume}}} \approx m_j\sqrt{|x_j|}.$$

The θ and "daily volume" have to be estimated in some way, i.e.

$$m_j = \frac{\theta}{\sqrt{\text{daily volume}}}$$

has to be estimated. The market impact term gives the cost as a fraction of daily traded volume ($|x_j|$). Therefore, the total cost when trading an amount x_j of asset j is given by

$$|x_j|(m_j|x_j|^{\frac{1}{2}}).$$

This leads us to the model:

$$\begin{array}{lrcl} \text{minimize} & f & & \\ \text{subject to} & \Sigma^{\frac{1}{2}}(w^0+x)-g & = & 0, \\ & \bar{r}^T(w^0+x) & = & t, \\ & e^Tx+e^Ty & = & 0, \\ & |x_j|(m_j|x_j|^{\frac{1}{2}}) & \leq & y_j, \\ & f \geq \|g\|\,. & & \end{array} \tag{9.40}$$

Now, defining the variable transformation

$$y_j = m_j\bar{y}_j$$

we obtain

$$\begin{array}{lrcl} \text{minimize} & f & & \\ \text{subject to} & \Sigma^{\frac{1}{2}}(w^0+x)-g & = & 0, \\ & \bar{r}^T(w^0+x) & = & t, \\ & e^Tx+m^T\bar{y} & = & 0, \\ & |x_j|^{3/2} & \leq & \bar{y}_j, \\ & f \geq \|g\|\,. & & \end{array} \tag{9.41}$$

As shown in Section 9.4.4.3 the set

$$|x_j|^{3/2} \leq \bar{y}_j$$

can be modelled by

$$\begin{array}{rcl} x_j & \leq & z_j, \\ -x_j & \leq & z_j, \\ z_j^2 & \leq & 2s_j\bar{y}_j, \\ u_j^2 & \leq & 2v_jq_j, \\ z_j & = & v_j, \\ s_j & = & u_j, \\ q_j & = & \frac{1}{8}, \\ q_j, s_j, \bar{y}_j, v_j, q_j & \geq & 0. \end{array} \tag{9.42}$$

9.7.2.4 Further reading

For further reading please see the reader to [18] in particular, and [21] and [1], which also contain relevant material.

Chapter 10

The optimizers for continuous problems

The most essential part of MOSEK is the optimizers. Each optimizer is designed to solve a particular class of problems i.e. linear, conic, or general nonlinear problems. The purpose of the present chapter is to discuss which optimizers are available for the continuous problem classes and how the performance of an optimizer can be tuned, if needed.

This chapter deals with the optimizers for *continuous problems* with no integer variables.

10.1 How an optimizer works

When the optimizer is called, it roughly performs the following steps:

Presolve: Preprocessing to reduce the size of the problem.

Dualizer: Choosing whether to solve the primal or the dual form of the problem.

Scaling: Scaling the problem for better numerical stability.

Optimize: Solving the actual optimization.

The first three preprocessing steps are transparent to the user, but useful to know about for tuning purposes. In general, the purpose of the preprocessing steps is to make the actual optimization more efficient and robust.

10.1.1 Presolve

Before an optimizer actually performs the optimization the problem is normally preprocessed using the so-called presolve. The purpose of the presolve is to

- remove redundant constraints,
- eliminate fixed variables,
- remove linear dependencies,

- substitute out free variables, and
- reduce the size of the optimization problem in general.

After the presolved problem has been optimized the solution is automatically postsolved so that the returned solution is valid for the original problem. Hence, the presolve is completely transparent. For further details about the presolve phase, please see [4, 5].

It is possible to fine-tune the behavior of the presolve or to turn it off entirely. If the presolve is known to be unable to reduce the size of a problem significantly, then turning off the presolve is beneficial. This is done by setting the parameter `MSK_IPAR_PRESOLVE_USE` to `MSK_PRESOLVE_MODE_OFF`.

The two most time-consuming steps of the presolve are usually

- the eliminator, and
- the linear dependency check.

Therefore, in some cases it is worthwhile to disable one or both of these.

The purpose of the eliminator is to eliminate free and implied free variables from the problem using substitution. For instance, given the constraints

$$\begin{array}{rcl} y & = & \sum_j x_j, \\ y, x & \geq & 0, \end{array}$$

y is an implied free variable that can be substituted out of the problem, if deemed worthwhile. By implied free variable is meant that the constraint $y \geq 0$ is redundant and hence y can be treated as a free variable.

For large scale problems the eliminator usually removes many constraints and variables. However, in some cases few or no eliminations can be performed and moreover, the eliminator may consume a lot of memory and time. If this is the case it is worthwhile to disable the eliminator by setting the parameter `MSK_IPAR_PRESOLVE_ELIMINATOR_USE` to `MSK_OFF`.

The purpose of the linear dependency check is to remove linear dependencies among the linear equalities. For instance, the three linear equalities

$$\begin{array}{rcl} x_1 + x_2 + x_3 & = & 1, \\ x_1 + 0.5x_2 & = & 0.5, \\ 0.5x_2 + x_3 & = & 0.5 \end{array}$$

contain exactly one linear dependency. This implies that one of the constraints can be dropped without changing the set of feasible solutions, i.e. one of the constraints is redundant. Removing linear dependencies is in general a good idea since it reduces the size of the problem. Moreover, the linear dependencies are likely to introduce numerical problems in the optimization phase, and therefore it is strongly recommended to build models without linear dependencies. In case the linear dependencies are removed at the modelling stage, the linear dependency check can safely be disabled by setting the parameter `MSK_IPAR_PRESOLVE_LINDEP_USE` to `MSK_OFF`.

10.1.2 Dualizer

It is well-known that all linear, conic, and convex optimization problems have an associated dual problem. Moreover, even if the dual instead of the primal problem is solved, it is possible to recover the solution to the original primal problem.

In general, it is very hard to say whether it is easier to solve the primal or the dual problem but MOSEK has some heuristics for deciding which of the two problems to solve. Which form of the problem (primal or dual) that is solved is displayed in the MOSEK log. Please note that the dualizer is transparent, and all solution values returned by the optimizer refer to the original primal problem.

The dualizer can be controlled manually by setting the parameters:

- `MSK_IPAR_INTPNT_SOLVE_FORM`: In case of the interior-point optimizer.

- `MSK_IPAR_SIM_SOLVE_FORM`: In case of the simplex optimizer.

Finally, please note that currently only linear problems may be dualized.

10.1.3 Scaling

Problems containing data with large and/or small coefficients, say 1.0e+9 or 1.0e-7, are often hard to solve. Significant digits might be truncated in calculations with finite precision, which can result in the optimizer relying on inaccurate calculations. Since computers work in finite precision, extreme coefficients should be avoided. In general, data around the same "order of magnitude" is preferred, and we will refer to a problem, satisfying this loose property, as being *well-scaled*. If the problem is not well scaled, MOSEK will try to scale (multiply) constraints and variables by suitable constants. MOSEK solves the scaled problem to improve the numerical properties.

The scaling process is transparent, i.e. the solution to the original problem is reported. It is important to be aware that the optimizer terminates when the termination criterion is met on the scaled problem, therefore significant primal or dual infeasibilities may occur after unscaling for badly scaled problems. The best solution to this problem is to reformulate it, making it better scaled.

By default MOSEK heuristically chooses a suitable scaling. The scaling for interior-point and simplex optimizers can be controlled with the parameters

`MSK_IPAR_INTPNT_SCALING` and `MSK_IPAR_SIM_SCALING`

respectively.

10.1.4 Using multiple CPU's

The interior-point optimizers in MOSEK have been parallelized. This means that if you solve linear, quadratic, conic, or general convex optimization problem using the interior-point optimizer, you can take advantage of multiple CPU's.

By default MOSEK uses one thread to solve the problem, but the number of threads (and thereby CPUs) employed can be changed by setting the parameter `MSK_IPAR_INTPNT_NUM_THREADS` This should never exceed the number of CPU's on the machine.

The speed-up obtained when using multiple CPUs is highly problem and hardware dependent, and consequently, it is advisable to compare single threaded and multi threaded performance for the given problem type to determine the optimal settings.

For small problems, using multiple threads will probably not be worthwhile.

Parameter name	Purpose
MSK_DPAR_INTPNT_TOL_PFEAS	Controls primal feasibility.
MSK_DPAR_INTPNT_TOL_DFEAS	Controls dual feasibility.
MSK_DPAR_INTPNT_TOL_REL_GAP	Controls relative gap.
MSK_DPAR_INTPNT_TOL_INFEAS	Controls when the problem is declared primal or dual infeasible.
MSK_DPAR_INTPNT_TOL_MU_RED	Controls when the complementarity is reduced enough.

Table 10.1: Parameters employed in termination criterion.

10.2 Linear optimization

10.2.1 Optimizer selection

For linear optimization problems two different types of optimizers are available. The default for linear problems is an interior-point optimizer, however, as an alternative the simplex optimizer can be employed.

The curious reader can consult [22] for a discussion about interior-point and simplex algorithms.

10.2.2 The interior-point optimizer

The MOSEK interior-point optimizer is an implementation of the homogeneous and self-dual algorithm. For a detailed description of the algorithm, please see [10].

10.2.2.1 Basis identification

It is well-known that an interior-point optimizer does not return an optimal basic solution unless the problem has a unique primal and dual optimal solution. Therefore, the interior-point optimizer has an optional post-processing step that computes an optimal basic solution starting from the optimal interior-point solution. More information about the basis identification procedure is found in [7].

Please note that a basic solution is often more accurate than an interior-point solution.

By default MOSEK performs a basis identification, however, if a basic solution is not needed, the basis identification procedure can be turned off. The parameters

- MSK_IPAR_INTPNT_BASIS,
- MSK_IPAR_BI_IGNORE_MAX_ITER, and
- MSK_IPAR_BI_IGNORE_NUM_ERROR

controls when basis identification is performed.

10.2.2.2 Interior-point termination criterion

The parameters in Table 10.1 control when the interior-point optimizer terminates.

10.2.3 The simplex based optimizer

An alternative to the interior-point optimizer is the simplex optimizer. The simplex optimizer employs a different approach than the interior-point optimizer when solving a problem. Contrary to the interior-point optimizer the simplex optimizer can exploit a guess for the optimal solution to reduce solution time. Depending on the problem it may be faster or slower to exploit a guess for the optimal solution. See Section 10.2.4 for a discussion.

MOSEK provides both a primal and a dual variant of the simplex optimizer — we will return to this later.

10.2.3.1 Simplex termination criterion

The simplex optimizer terminates when it finds an optimal basic solution or an infeasibility certificate. A basic solution is optimal when it is primal and dual feasible, see (9.1) and (9.2) for a definition of the primal and dual problem. Due the fact that to computations are performed in finite precision MOSEK allows violation of primal and dual feasibility within certain tolerances. The user can control the allowed primal and dual infeasibility with the parameters `MSK_DPAR_BASIS_TOL_X` and `MSK_DPAR_BASIS_TOL_S`.

10.2.3.2 Starting from an existing solution

When using the simplex optimizer it may be possible to reuse an existing solution and thereby reduce the solution time significantly. When a simplex optimizer starts from an existing solution it is said to perform a “hot-start”. If the user is solving a sequence of optimization problems by solving the problem, making modifications, and solving again, MOSEK will hot-start automatically.

Setting the parameter `MSK_IPAR_OPTIMIZER` to `MSK_OPTIMIZER_FREE_SIMPLEX` instructs MOSEK to select automatically between the primal and the dual simplex optimizers. Hence, MOSEK tries to choose the best optimizer given the problem and the available solution.

By default MOSEK uses presolve when performing a hot-start. If the optimizer only needs very few iterations to find the optimal solution it may be better to turn off the presolve.

10.2.3.3 Numerical difficulties in the simplex optimizers

MOSEK is designed to minimize numerical difficulties, however, in rate cases the optimizer may have a hard time solving a problem. MOSEK counts a numerical unexpected behavior inside the optimizer as a “set-back”. The user can define how many set-backs the optimizer accepts, and if that number is exceeded, the optimization will be aborted. Set-Backs are implemented to avoid long sequences where the optimizer tries to recover from an unstable situation.

What counts as a set-back? It is hard to say without getting very technical but obvious cases are repeated singularities when factorizing the basis matrix, repeated loss of feasibility, degeneracy problems (no progress in objective) or other events indicating numerical difficulties. If the simplex optimizer encounters a lot of set-backs the problem is usually badly scaled. In such a situation try to reformulate into a better scaled problem. If a lot of set-backs still occur, then trying one or more of the following suggestions may be worthwhile.

- Raise tolerances for allowed primal or dual feasibility: Hence, increase the value of
  - `MSK_DPAR_BASIS_TOL_X`, and

  - MSK_DPAR_BASIS_TOL_S.

- Raise or lower pivot tolerance: Change the MSK_DPAR_SIMPLEX_ABS_TOL_PIV parameter.
- Switch optimizer: Try another optimizer.
- Switch off crash: Set both MSK_IPAR_SIM_PRIMAL_CRASH and MSK_IPAR_SIM_DUAL_CRASH to 0.
- Experiment with other pricing strategies: Try different values for the parameters
  - MSK_IPAR_SIM_PRIMAL_SELECTION and
  - MSK_IPAR_SIM_DUAL_SELECTION.
- If you are using hot-starts, in rare cases switching off this feature may improve stability. This is controlled by the MSK_IPAR_SIM_HOTSTART parameter.
- Increase maximum set-backs allowed controlled by MSK_IPAR_SIM_MAX_NUM_SETBACKS.
- If the problem repeatedly becomes infeasible try switching off the special degeneracy handling. See the parameter MSK_IPAR_SIM_DEGEN for details.

10.2.4 The interior-point or the simplex optimizer?

Given a linear optimization problem, which optimizer is the best: The primal simplex, the dual simplex or the interior-point optimizer?

It is impossible to provide a general answer to this question, however, the interior-point optimizer behaves more predictably — it tends to use between 20 and 100 iterations, almost independently of problem size — but cannot perform hot-start, while simplex can take advantage of an initial solution, but is less predictable for cold-start. The interior-point optimizer is used by default.

10.2.5 The primal or the dual simplex variant?

MOSEK provides both a primal and a dual simplex optimizer. Predicting which simplex optimizer is faster is simply impossible, however, in recent years the dual optimizer has experienced several algorithmic and computational improvements, which, in our experience, makes it faster on average than the primal simplex optimizer. Still, it depends much on the problem structure and size.

Setting the MSK_IPAR_OPTIMIZER parameter to MSK_OPTIMIZER_FREE_SIMPLEX instructs MOSEK to choose which simplex optimizer to use automatically.

To summarize, if you want to know which optimizer is faster for a given problem type, you should try all the optimizers.

10.3 Linear network optimization

10.3.1 Network flow problems

Linear optimization problems with the network flow structure specified in Section 9.2 can in most cases be solved significantly faster with a specialized version of the simplex method [2], rather than with the general solvers.

Parameter name	Purpose
MSK_DPAR_INTPNT_CO_TOL_PFEAS	Controls primal feasibility
MSK_DPAR_INTPNT_CO_TOL_DFEAS	Controls dual feasibility
MSK_DPAR_INTPNT_CO_TOL_REL_GAP	Controls relative gap
MSK_DPAR_INTPNT_TOL_INFEAS	Controls when the problem is declared infeasible
MSK_DPAR_INTPNT_CO_TOL_MU_RED	Controls when the complementarity is reduced enough

Table 10.2: Parameters employed in termination criterion.

MOSEK includes a network simplex solver, which usually solves network problems 10 to 100 times faster than the standard simplex optimizers implemented by MOSEK.

To use the network simplex optimizer, do the following

- Input the network flow problem as an ordinary linear optimization problem.
- Set
  - the MSK_IPAR_SIM_NETWORK_DETECT parameter to 0, and
  - the MSK_IPAR_OPTIMIZER parameter to MSK_OPTIMIZER_FREE_SIMPLEX.
- Optimize the problem.

MOSEK will automatically detect the network structure and apply the specialized simplex optimizer.

10.3.2 Embedded network problems

Often problems contains both large parts with network structure and some non-network constraints or variables — such problems are said to have *embedded network structure*. If the procedure described above is applied, MOSEK will try to exploit this structure to speed up the optimization.

This is done by heuristically detecting the largest network embedded in the problem, solving this using the network simplex optimizer, and using this solution to hot-start a normal simplex optimizer.

The MSK_IPAR_SIM_NETWORK_DETECT parameter defines how large a percentage of the problem should be a network before the specialized solver is applied. In general, it is recommended to use the network optimizer only on problems containing a substantial embedded network.

10.4 Conic optimization

10.4.1 The interior-point optimizer

For conic optimization problems only an interior-point type optimizer is available. The interior-point optimizer is an implementation of the so-called homogeneous and self-dual algorithm. For a detailed description of the algorithm, please see [6].

10.4.1.1 Interior-point termination criteria

The parameters controlling when the conic interior-point optimizer terminates are shown in Table 10.2.

Parameter name	Purpose
MSK_DPAR_INTPNT_NL_TOL_PFEAS	Controls primal feasibility
MSK_DPAR_INTPNT_NL_TOL_DFEAS	Controls dual feasibility
MSK_DPAR_INTPNT_NL_TOL_REL_GAP	Controls relative gap
MSK_DPAR_INTPNT_TOL_INFEAS	Controls when the problem is declared infeasible
MSK_DPAR_INTPNT_NL_TOL_MU_RED	Controls when the complementarity is reduced enough

Table 10.3: Parameters employed in termination criteria.

10.5 Nonlinear convex optimization

10.5.1 The interior-point optimizer

For quadratic, quadratically constrained, and general convex optimization problems only an interior-point type optimizer is available. The interior-point optimizer is an implementation of the homogeneous and self-dual algorithm. For a detailed description of the algorithm, please see [8, 9].

10.5.1.1 Interior-point termination criteria

The parameters controlling when the general convex interior-point optimizer terminates are shown in Table 10.3.

10.6 Solving problems in parallel

If a computer has multiple CPUs, or has a CPU with multiple cores, it is possible for MOSEK to take advantage of this to speed up solution times.

10.6.1 Thread safety

The MOSEK API is thread-safe provided that a task is only modified or accessed from one thread at any given time — accessing two separate tasks from two separate threads at the same time is safe. Sharing an environment between threads is safe.

10.6.2 The parallelized interior-point optimizer

The interior-point optimizer is capable of using multiple CPUs or cores. This implies that whenever the MOSEK interior-point optimizer solves an optimization problem, it will try to divide the work so that each CPU gets a share of the work. The user decides how many CPUs MOSEK should exploit.

It is not always possible to divide the work equally, and often parts of the computations and the coordination of the work is processed sequentially, even if several CPUs are present. Therefore, the speed-up obtained when using multiple CPUs is highly problem dependent. However, as a rule of thumb, if the problem solves very quickly, i.e. in less than 60 seconds, it is not advantageous to use the parallel option.

The MSK_IPAR_INTPNT_NUM_THREADS parameter sets the number of threads (and therefore the number of CPUs) that the interior point optimizer will use.

Optimizer	Associated parameter	Default priority
MSK_OPTIMIZER_INTPNT	MSK_IPAR_CONCURRENT_PRIORITY_INTPNT	4
MSK_OPTIMIZER_FREE_SIMPLEX	MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX	3
MSK_OPTIMIZER_PRIMAL_SIMPLEX	MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX	2
MSK_OPTIMIZER_DUAL_SIMPLEX	MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX	1

Table 10.4: Default priorities for optimizer selection in concurrent optimization.

10.6.3 The concurrent optimizer

An alternative to the parallel interior-point optimizer is the *concurrent optimizer*. The idea of the concurrent optimizer is to run multiple optimizers on the same problem concurrently, for instance, it allows you to apply the interior-point and the dual simplex optimizers to a linear optimization problem concurrently. The concurrent optimizer terminates when the first of the applied optimizers has terminated successfully, and it reports the solution of the fastest optimizer. In that way a new optimizer has been created which essentially performs as the fastest of the interior-point and the dual simplex optimizers.Hence, the concurrent optimizer is the best one to use if there are multiple optimizers available in MOSEK for the problem and you cannot say beforehand which one will be faster.

Note in particular that any solution present in the task will also be used for hot-starting the simplex algorithms. One possible scenario would therefore be running a hot-start dual simplex in parallel with interior point, taking advantage of both the stability of the interior-point method and the ability of the simplex method to use an initial solution.

By setting the

```
MSK_IPAR_OPTIMIZER
```

parameter to

```
MSK_OPTIMIZER_CONCURRENT
```

the concurrent optimizer chosen.

The number of optimizers used in parallel is determined by the

```
MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS.
```

parameter. Moreover, the optimizers are selected according to a preassigned priority with optimizers having the highest priority being selected first. The default priority for each optimizer is shown in Table 10.6.3. For example, setting the `MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS` parameter to 2 tells the concurrent optimizer to the apply the two optimizers with highest priorities: In the default case that means the interior-point optimizer and one of the simplex optimizers.

10.6.3.1 Concurrent optimization from the command line

The command line

```
  mosek afiro.mps  -d MSK_IPAR_OPTIMIZER MSK_OPTIMIZER_CONCURRENT \
       -d MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS 2
```

produces the following (edited) output:

```
...

Number of concurrent optimizers          : 2
Optimizer selected for thread number 0   : interior-point (threads = 1)
Optimizer selected for thread number 1   : free simplex
Total number of threads required         : 2

...

Thread number 1 (free simplex) terminated first.

...

Concurrent optimizer terminated. CPU Time: 0.03. Real Time: 0.00.
```

As indicated in the log information, the interior-point and the free simplex optimizers are employed concurrently. However, only the output from the optimizer having the highest priority is printed to the screen. In the example this is the interior-point optimizer.

The line

```
Total number of threads required         : 2
```

indicates the number of threads used. If the concurrent optimizer should be effective, this should be lower than the number of CPUs.

In the above example the simplex optimizer finishes first as indicated in the log information.

Chapter 11

The optimizer for mixed integer problems

A problem is a mixed integer optimization problem when one or more of the variables are constrained to be integers. The integer optimizer available in MOSEK can solve integer optimization problems involving

- linear,
- quadratic and
- conic

constraints. However, a problem is not allowed to have both conic constraints and quadratic objective or constraints.

Readers unfamiliar with integer optimization are strongly recommended to consult some relevant literature, e.g. the book [25] by Wolsey is a good introduction to integer optimization.

11.1 Some notation

In general, an integer optimization problem has the form

$$\begin{array}{rclcccl} z^* & = & \text{minimize} & & c^T x & & \\ & & \text{subject to} & l^c \leq & Ax & \leq & u^c, \\ & & & l^x \leq & Ax & \leq & u^x, \\ & & & & x_j \in \mathcal{Z}, & & \forall j \in \mathcal{J}, \end{array} \tag{11.1}$$

where $\mathcal{J}$ is an index set specifying which variables are integer constrained. Frequently we talk about the continuous relaxation of an integer optimization problem defined as

$$\begin{array}{rclcccl} \underline{z} & = & \text{minimize} & & c^T x & & \\ & & \text{subject to} & l^c \leq & Ax & \leq & u^c, \\ & & & l^x \leq & Ax & \leq & u^x \end{array} \tag{11.2}$$

i.e. we ignore the constraint

$$x_j \in \mathcal{Z}, \ \forall j \in \mathcal{J}.$$

Moreover, let $\hat{x}$ be any feasible solution to (11.1) and define

$$\overline{z} := c^T \hat{x}.$$

It should be obvious that

$$\underline{z} \leq z^* \leq \overline{z}$$

holds. This is an important observation since if we assume that it is not possible to solve the mixed integer optimization problem within a reasonable time frame, but that a feasible solution can be found, then the natural question is: How far is the *obtained* solution from the *optimal* solution? The answer is that no feasible solution can have an objective value smaller than $\underline{z}$, which implies that the obtained solution is no further away from the optimum than $\overline{z} - \underline{z}$.

11.2 An important fact about integer optimization problems

It is important to understand that in a worst-case scenario, the time required to solve integer optimization problems grows exponentially with the size of the problem. For instance, assume that a problem contains n binary variables, then the time required to solve the problem in the worst case may be proportional to 2^n. It is a simple exercise to verify that 2^n is huge even for moderate values of n.

In practice this implies that the focus should be on computing a near optimal solution quickly rather than at locating an optimal solution.

11.3 How the integer optimizer works

The process of solving an integer optimization problem can be split in three phases:

Presolve: In this phase the optimizer tries to reduce the size of the problem using preprocessing techniques. Moreover, it strengthens the continuous relaxation, if possible.

Heuristic: Using heuristics the optimizer tries to guess a good feasible solution.

Optimization: The optimal solution is located using a variant of the branch-and-cut method.

In some cases the integer optimizer may locate an optimal solution in the preprocessing stage or conclude that the problem is infeasible. Therefore, the heuristic and optimization stages may never be performed.

11.3.1 Presolve

In the preprocessing stage redundant variables and constraints are removed. The presolve stage can be turned off using the `MSK_IPAR_MIO_PRESOLVE_USE` parameter .

11.3.2 Heuristic

Initially, the integer optimizer tries to guess a good feasible solution using different heuristics:

- First a very simple rounding heuristic is employed.
- Next, if deemed worthwhile, the *feasibility pump* heuristic is used.
- Finally, if the two previous stages did not produce a good initial solution, more sophisticated heuristics are used.

The following parameters can be used to control the effort made by the integer optimizer to find an initial feasible solution.

- `MSK_IPAR_MIO_HEURISTIC_LEVEL`: Controls how sophisticated and computationally expensive a heuristic to employ.
- `MSK_DPAR_MIO_HEURISTIC_TIME`: The minimum amount of time to spend in the heuristic search.
- `MSK_IPAR_MIO_FEASPUMP_LEVEL`: Controls how aggressively the feasibility pump heuristic is used.

11.3.3 The optimization phase

This phase solves the problem using the branch and cut algorithm.

11.4 Termination criterion

In general, it is impossible to find an exact feasible and optimal solution to an integer optimization problem in a reasonable amount of time, though in many practical cases it may be possible. Therefore, the integer optimizer employs a relaxed feasibility and optimality criterion to determine when a satisfactory solution is located.

A candidate solution, i.e. a solution to (11.2), is said to be an integer feasible solution if the criterion

$$\min(|x_j| - \lfloor x_j \rfloor, \lceil x_j \rceil - |x_j|) \leq \max(\delta_1, \delta_2 |x_j|) \ \forall j \in \mathcal{J}$$

is satisfied. Hence, such a solution is defined as a feasible solution to (11.1).

Whenever the integer optimizer locates an integer feasible solution it will check if the criterion

$$\overline{z} - \underline{z} \leq \max(\delta_3, \delta_4 \max(1, |\overline{z}|))$$

is satisfied. If this is the case, the integer optimizer terminates and reports the integer feasible solution as an optimal solution. Please note that $\underline{z}$ is a valid lower bound determined by the integer optimizer during the solution process, i.e.

$$\underline{z} \leq z^*.$$

The lower bound $\underline{z}$ normally increases during the solution process.

The δ tolerances can are specified using parameters — see Table 11.1. If an optimal solution cannot be located within a reasonable time, it may be advantageous to employ a relaxed termination criterion after some time. Whenever the integer optimizer locates an integer feasible solution and has spent at

Tolerance	Parameter name
δ_1	MSK_DPAR_MIO_TOL_ABS_RELAX_INT
δ_2	MSK_DPAR_MIO_TOL_REL_RELAX_INT
δ_3	MSK_DPAR_MIO_TOL_ABS_GAP
δ_4	MSK_DPAR_MIO_TOL_REL_GAP
δ_5	MSK_DPAR_MIO_NEAR_TOL_ABS_GAP
δ_6	MSK_DPAR_MIO_NEAR_TOL_REL_GAP

Table 11.1: Integer optimizer tolerances.

Parameter name	Delayed	Explanation
MSK_IPAR_MIO_MAX_NUM_BRANCHES	Yes	Maximum number of branches allowed.
MSK_IPAR_MIO_MAX_NUM_RELAXS	Yes	Maximum number of relaxations allowed.

Table 11.2: Parameters affecting the termination of the integer optimizer.

least the number of seconds defined by the `MSK_DPAR_MIO_DISABLE_TERM_TIME` parameter on solving the problem, it will check whether the criterion

$$\overline{z} - \underline{z} \leq \max(\delta_5, \delta_6 \max(1, |\overline{z}|))$$

is satisfied. If it is satisfied, the optimizer will report that the candidate solution is **near optimal** and then terminate. All δ tolerances can be adjusted using suitable parameters — see Table 11.1. In Table 11.2 some other parameters affecting the integer optimizer termination criterion are shown. Please note that if the effect of a parameter is delayed, the associated termination criterion is applied only after some time, specified by the `MSK_DPAR_MIO_DISABLE_TERM_TIME` parameter.

11.5 How to speed up the solution process

As mentioned previously, in many cases it is not possible to find an optimal solution to an integer optimization problem in a reasonable amount of time. Some suggestions to reduce the solution time are:

- Relax the termination criterion: In case the run time is not acceptable, the first thing to do is to relax the termination criterion — see Section 11.4 for details.
- Specify a good initial solution: In many cases a good feasible solution is either known or easily computed using problem specific knowledge. If a good feasible solution is known, it is usually worthwhile to use this as a starting point for the integer optimizer.
- Improve the formulation: A mixed integer optimization problem may be impossible to solve in one form and quite easy in another form. However, it is beyond the scope of this manual to discuss good formulations for mixed integer problems. For discussions on this topic see for example [25].

Chapter 12

Analyzing infeasible problems

When developing and implementing a new optimization model, the first attempts will often be either infeasible, due to specification of inconsistent constraints, or unbounded, if important constraints have been left out.

In this chapter we will

- go over an example demonstrating how to locate infeasible constraints using the MOSEK infeasibility report tool,
- discuss in more general terms which properties that may cause infeasibilities, and
- present the more formal theory of infeasible and unbounded problems.

12.1 Example: Primal infeasibility

A problem is said to be *primal infeasible* if no solution exists that satisfy all the constraints of the problem.

As an example of a primal infeasible problem consider the problem of minimizing the cost of transportation between a number of production plants and stores: Each plant produces a fixed number of goods, and each store has a fixed demand that must be met. Supply, demand and cost of transportation per unit are given in figure 12.1.

The problem represented in figure 12.1 is infeasible, since the total demand

$$2300 = 1100 + 200 + 500 + 500 \tag{12.1}$$

exceeds the total supply

$$2200 = 200 + 1000 + 1000 \tag{12.2}$$

If we denote the number of transported goods from plant i to store j by x_{ij}, the problem can be

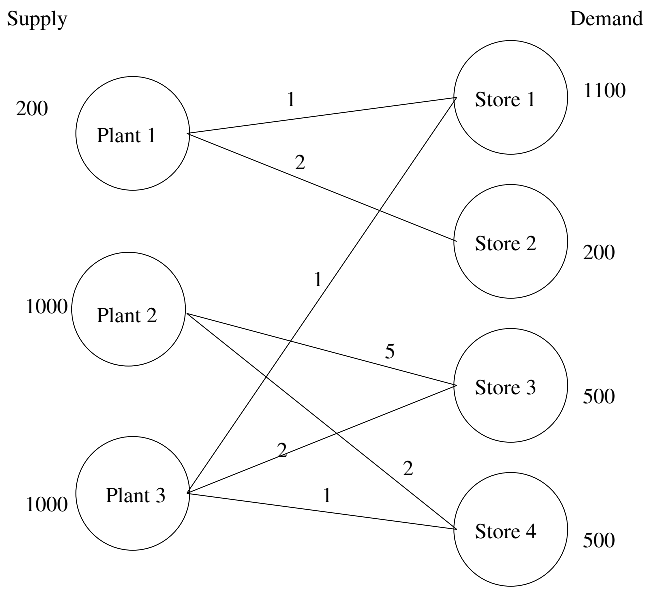

Figure 12.1: Supply, demand and cost of transportation.

formulated as the LP:

$$
\begin{array}{lccccccccccccccc}
\text{minimize} & x_{11} & + & 2x_{12} & + & 5x_{23} & + & 2x_{24} & + & x_{31} & + & 2x_{33} & + & x_{34} & & \\
\text{subject to} & x_{11} & + & x_{12} & & & & & & & & & & & \leq & 200, \\
 & & & & & x_{23} & + & x_{24} & & & & & & & \leq & 1000, \\
 & & & & & & & & & x_{31} & + & x_{33} & + & x_{34} & \leq & 1000, \\
 & x_{11} & & & & & & & + & x_{31} & & & & & = & 1100, \\
 & & & x_{12} & & & & & & & & & & & = & 200, \\
 & & & & & x_{23} & + & & & & & x_{33} & & & = & 500, \\
 & & & & & & & x_{24} & + & & & & & x_{34} & = & 500, \\
 & x_{ij} \geq 0. & & & & & & & & & & & & & &
\end{array}
\tag{12.3}
$$

Solving the problem (12.3) using MOSEK will result in a solution, a solution status and a problem status. Among the log output from the execution of MOSEK on the above problem are the lines:

```
Basic solution
Problem status  : PRIMAL_INFEASIBLE
Solution status : PRIMAL_INFEASIBLE_CER
```

The first line indicates that the problem status is primal infeasible. The second line says that a *certificate of the infeasibility* was found. The certificate is returned in place of the solution to the problem.

12.1.1 Locating the cause of primal infeasibility

Usually a primal infeasible problem status is caused by a mistake in formulating the problem and therefore the question arises: "What is the cause of the infeasible status?" When trying to answer this question, it is often advantageous to follow these steps:

- Remove the objective function. This does not change the infeasible status but simplifies the problem, eliminating any possibility of problems related to the objective function.

- Consider whether your problem has some necessary conditions for feasibility and examine if these are satisfied, e.g. total supply should be greater than or equal to total demand.

- Verify that coefficients and bounds are reasonably sized in your problem.

If the problem is still primal infeasible, some of the constraints must be relaxed or removed completely. The MOSEK infeasibility report (Section 12.1.3) may assist you in finding the constraints causing the infeasibility.

Possible ways of relaxing your problem include:

- Increasing (decreasing) upper (lower) bounds on variables and constraints.

- Removing suspected constraints from the problem.

Returning to the transportation example, we discover that removing the fifth constraint

$$x_{12} = 200 \tag{12.4}$$

makes the problem feasible.

12.1.2 Locating the cause of dual infeasibility

A problem may also be *dual infeasible*. In this case the primal problem is often unbounded, mening that feasbile solutions exists such that the objective tends towards infinity. An example of a dual infeasible and primal unbounded problem is:

$$\begin{array}{ll} \text{minimize} & x_1 \\ \text{subject to} & x_1 \leq 5 \end{array} \tag{12.5}$$

To resolve a dual infeasibility the primal problem must be made more restricted by

- Adding upper or lower bounds on variables or constraints.

- Removing variables.

- Changing the objective.

12.1.2.1 A cautious note

The problem

$$\begin{array}{ll} \text{minimize} & 0 \\ \text{subject to} & 0 \leq x_1, \\ & x_j \leq x_{j+1}, \quad j = 1, \ldots, n-1, \\ & x_n \leq -1 \end{array} \tag{12.6}$$

is clearly infeasible. Moreover, if any one of the constraints are dropped, then the problem becomes feasible.

This illustrates the worst case scenario that all, or at least a significant portion, of the constraints are involved in the infeasibility. Hence, it may not always be easy or possible to pinpoint a few constraints which are causing the infeasibility.

12.1.3 The infeasibility report

MOSEK includes functionality for diagnosing the cause of a primal or a dual infeasibility. It can be turned on by setting the `MSK_IPAR_INFEAS_REPORT_AUTO` to `MSK_ON`. This causes MOSEK to print a report on variables and constraints involved in the infeasibility.

The `MSK_IPAR_INFEAS_REPORT_LEVEL` parameter controls the amount of information presented in the infeasibility report. The default value is 1.

12.1.3.1 Example: Primal infeasibility

We will reuse the example (12.3) located in `infeas.lp`:

```
\
\ An example of an infeasible linear problem.
\
minimize
 obj: + 1 x11 + 2 x12 + 1 x13
      + 4 x21 + 2 x22 + 5 x23
      + 4 x31 + 1 x32 + 2 x33
st
  s0: + x11 + x12      <= 200
  s1: + x23 + x24      <= 1000
  s2: + x31 +x33 + x34 <= 1000
  d1: + x11 + x31       = 1100
  d2: + x12             = 200
  d3: + x23 + x33       = 500
  d4: + x24 + x34       = 500
bounds
end
```

Using the command line

```
mosek -d MSK_IPAR_INFEAS_REPORT_AUTO MSK_ON infeas.lp
```

MOSEK produces the following infeasibility report

```
MOSEK PRIMAL INFEASIBILITY REPORT.

Problem status: The problem is primal infeasible

The following constraints are involved in the primal infeasibility.

Index    Name      Lower bound      Upper bound      Dual lower       Dual upper
0        s0        NONE             2.000000e+002    0.000000e+000    1.000000e+000
2        s2        NONE             1.000000e+003    0.000000e+000    1.000000e+000
3        d1        1.100000e+003    1.100000e+003    1.000000e+000    0.000000e+000
4        d2        2.000000e+002    2.000000e+002    1.000000e+000    0.000000e+000

The following bound constraints are involved in the infeasibility.

Index    Name      Lower bound      Upper bound      Dual lower       Dual upper
8        x33       0.000000e+000    NONE             1.000000e+000    0.000000e+000
10       x34       0.000000e+000    NONE             1.000000e+000    0.000000e+000
```

The infeasibility report is divided into two sections where the first section shows which constraints that are important for the infeasibility. In this case the important constraints are the ones named `s0`, `s2`, `d1`, and `d2`. The values in the columns "`Dual lower`" and "`Dual upper`" are also useful,since a non-zero *dual lower* value for a constraint implies that the lower bound on the constraint is important for the infeasibility. Similarly, a non-zero *dual upper* value implies that the upper bound on the constraint is important for the infeasibility.

It is also possible to obtain the infeasible subproblem. The executing the command

```
mosek -d MSK_IPAR_INFEAS_REPORT_AUTO MSK_ON infeas.lp -info rinfeas.lp
```

produces the files `rinfeas.bas.inf.lp`. In this case the content of the file `rinfeas.bas.inf.lp` is

```
minimize
 Obj: + CFIXVAR
st
 s0: + x11 + x12 <= 200
 s2: + x31 + x33 + x34 <= 1e+003
 d1: + x11 + x31 = 1.1e+003
 d2: + x12 = 200
bounds
 x11 free
 x12 free
 x13 free
 x21 free
 x22 free
 x23 free
 x31 free
 x32 free
 x24 free
 CFIXVAR = 0e+000
end
```

which is an optimization problem. Please note that this optimization problem is identical to (12.3), except that the objective and some of the constraints and bounds have been removed. Executing the command

```
mosek -d MSK_IPAR_INFEAS_REPORT_AUTO MSK_ON rinfeas.bas.inf.lp
```

demonstrates that the reduced problem is **primal infeasible**. However, since the reduced problem is usually smaller, it should be easier to locate the cause of the infeasibility in this rather than in the original problem (12.3).

12.1.3.2 Example: Dual infeasibility

The example problem

```
minimize -  200 y1 - 1000 y2 - 1000 y3
         - 1100 y4 -  200 y5 -  500 y6
         -  500 y7
subject to
   x11: y1+y4 < 1
   x12: y1+y5 < 2
   x23: y2+y6 < 5
   x24: y2+y7 < 2
   x31: y3+y4 < 1
   x33: y3+y6 < 2
   x44: y3+y7 < 1
bounds
   y1 < 0
   y2 < 0
   y3 < 0
   y4 free
   y5 free
   y6 free
   y7 free
end
```

is dual infeasible. This can be verified by proving that

```
y1=-1, y2=-1, y3=0, y4=1, y5=1
```

is a certificate of dual infeasibility. In this example the following infeasibility report is produced (slightly edited):

```
he following constraints are involved in the infeasibility.

Index     Name              Activity         Objective        Lower bound      Upper bound
0         x11               -1.000000e+00                     NONE             1.000000e+00
4         x31               -1.000000e+00                     NONE             1.000000e+00
```

```
The following variables are involved in the infeasibility.

Index    Name             Activity         Objective        Lower bound      Upper bound
3        y4               -1.000000e+00    -1.100000e+03    NONE             NONE
Interior-point solution
Problem status  : DUAL_INFEASIBLE
Solution status : DUAL_INFEASIBLE_CER
Primal - objective: 1.1000000000e+03   eq. infeas.: 0.00e+00 max bound infeas.: 0.00e+00 cone infeas.: 0.00e+00
Dual   - objective: 0.0000000000e+00   eq. infeas.: 0.00e+00 max bound infeas.: 0.00e+00 cone infeas.: 0.00e+00
```

Let x^* denote the reported primal solution. MOSEK states

- that the problem is *dual infeasible*,
- that the reported solution is a certificate of dual infeasibility, and
- that the infeasibility measure for x^* is approximately zero.

Since it was an maximization problem, this implies that

$$c^t x^* > 0. \tag{12.7}$$

For a minimization problem this inequality would have been reversed — see (12.19).

From the infeasibility report we see that the variable `y4`, and the constraints `x11` and `x33` are involved in the infeasibility since these appear with non-zero values in the "`Activity`" column.

One possible strategy to "fix" the infeasibility is to modify the problem so that the certificate of infeasibility becomes invalid. In this case we might do one the the following things:

- Put a lower bound in `y3`. This will directly invalidate the certificate of dual infeasibility.
- Increase the object coefficient of `y3`. Changing the coefficients sufficiently will invalidate the inequality (12.7) and thus the certificate.
- Put lower bounds on `x11` or `x31`. This will directly invalidate the certificate of infeasibility.

Please note that modifying the problem to invalidate the reported certificate does *not* imply that the problem becomes dual feasible — the infeasibility may simply "move", resulting in a new infeasibility.

More often, the reported certificate can be used to give a hint about errors or inconsistencies in the model that produced the problem.

12.2 Theory concerning infeasible problems

This section discusses the theory of infeasibility certificates and how MOSEK uses a certificate to produce an infeasibility report. In general, MOSEK solves the problem

$$\begin{array}{lccccc} \text{minimize} & & & c^T x + c^f & & \\ \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x \end{array} \tag{12.8}$$

where the corresponding dual problem is

$$\begin{array}{lll} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c & \\ & +(l^x)^T s_l^x - (u^x)^T s_u^x + c^f & \\ \text{subject to} & A^T y + s_l^x - s_u^x = c, & \\ & -y + s_l^c - s_u^c = 0, & \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0. & \end{array} \tag{12.9}$$

We use the convension that for any bound that is not finite, the corresponding dual variable is fixed at zero (and thus will have no influence on the dual problem). For example

$$l_j^x = -\infty \;\Rightarrow\; (s_l^x)_j = 0 \tag{12.10}$$

12.2.1 Certificat of primal infeasibility

A certificate of primal infeasibility is *any* solution to the homogenized dual problem

$$\begin{array}{lll} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c & \\ & +(l^x)^T s_l^x - (u^x)^T s_u^x & \\ \text{subject to} & A^T y + s_l^x - s_u^x = 0, & \\ & -y + s_l^c - s_u^c = 0, & \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0. & \end{array} \tag{12.11}$$

with a positive objective value. That is, $(s_l^{c*}, s_u^{c*}, s_l^{x*}, s_u^{x*})$ is a certificat of primal infeasibility if

$$(l^c)^T s_l^{c*} - (u^c)^T s_u^{c*} + (l^x)^T s_l^{x*} - (u^x)^T s_u^{x*} > 0 \tag{12.12}$$

and

$$\begin{array}{lll} A^T y + s_l^{x*} - s_u^{x*} & = & 0, \\ -y + s_l^{c*} - s_u^{c*} & = & 0, \\ s_l^{c*}, s_u^{c*}, s_l^{x*}, s_u^{x*} \geq 0. & & \end{array} \tag{12.13}$$

The well-known Farkas Lemma tells us that (12.8) is infeasible if and only if a certificat of primal infeasibility exists.

Let $(s_l^{c*}, s_u^{c*}, s_l^{x*}, s_u^{x*})$ be a certificate of primal infeasibility then

$$(s_l^{c*})_i > 0 \quad ((s_u^{c*})_i > 0) \tag{12.14}$$

implies that the lower (upper) bound on the ith constraint is important for the infeasibility. Furthermore,

$$(s_l^{x*})_j > 0 \quad ((s_u^{x*})_i > 0) \tag{12.15}$$

implies that the lower (upper) bound on the jth variable is important for the infeasibility.

12.2.2 Certificat of dual infeasibility

A certificate of dual infeasibility is *any* solution to the problem

$$\begin{array}{llllll} \text{minimize} & & & c^T x & & \\ \text{subject to} & \bar{l}^c & \leq & Ax & \leq & \bar{u}^c, \\ & \bar{l}^x & \leq & x & \leq & \bar{u}^x \end{array} \tag{12.16}$$

with negative objective value, where we use the definitions

$$\bar{l}_i^c := \begin{cases} 0, & l_i^c > -\infty, \\ -\infty, & \text{otherwise}, \end{cases} \qquad \bar{u}_i^c := \begin{cases} 0, & u_i^c < \infty, \\ \infty, & \text{otherwise}, \end{cases} \tag{12.17}$$

and

$$\bar{l}_i^x := \begin{cases} 0, & l_i^x > -\infty, \\ -\infty, & \text{otherwise}, \end{cases} \quad \text{and } \bar{u}_i^x := \begin{cases} 0, & u_i^x < \infty, \\ \infty, & \text{otherwise}. \end{cases} \tag{12.18}$$

Stated differently, a certificate of dual infeasibility is any x^* such that

$$\begin{array}{ccccc} & & c^T x^* & < & 0, \\ \bar{l}^c & \leq & A x^* & \leq & \bar{u}^c, \\ \bar{l}^x & \leq & x^* & \leq & \bar{u}^x \end{array} \tag{12.19}$$

The well-known Farkas Lemma tells us that (12.9) is infeasible if and only if a certificat of dual infeasibility exists.

Observe that if x^* is a certificate of dual infeasibility then for any j such that

$$x_j^* \neq 0, \tag{12.20}$$

variable j is involved in the dual infeasibility.

Chapter 13

Feasibility repair

Section 12.1.1 discusses how MOSEK treats infeasible problems. In particular, it is discussed which information MOSEK returns when a problem is infeasible and how this information can be used to pinpoint the elements causing the infeasibility.

In this section we will discuss a method for repairing a primal infeasible problem by relaxing the constraints in a controlled way. For the sake of simplicity we discuss the method in the context of linear optimization. MOSEK can also repair infeasibilities in quadratic and conic optimization problems possibly having integer constrained variables. Please note that infeasibilities in nonlinear optimization problems can't be repaired using the method described below.

13.1 The main idea

Consider the linear optimization problem with m constraints and n variables

$$\begin{array}{lccccc} \text{minimize} & & & c^T x + c^f & & \\ \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \end{array} \tag{13.1}$$

which we assume is infeasible. Moreover, we assume that

$$(l^c)_i \leq (u^c)_i, \ \forall i \tag{13.2}$$

and

$$(l^x)_j \leq (u^x)_j, \ \forall j \tag{13.3}$$

because otherwise the problem (13.1) is trivially infeasible.

One way of making the problem feasible is to reduce the lower bounds and increase the upper bounds. If the change is sufficiently large the problem becomes feasible.

One obvious question is: What is the smallest change to the bounds that will make the problem feasible?

We associate a weight with each bound:

- $w_l^c \in R^m$ (associated with l^c),

- $w_u^c \in R^m$ (associated with u^c),
- $w_l^x \in R^n$ (associated with l^x),
- $w_u^x \in R^n$ (associated with u^x),

Now, the problem

$$\begin{array}{lcccl} \text{minimize} & & p & & \\ \text{subject to} \quad l^c & \leq & Ax + v_l^c - v_u^c & \leq & u^c, \\ l^x & \leq & x + v_l^x - v_u^x & \leq & u^x, \\ & & (w_l^c)^T v_l^c + (w_u^c)^T v_u^c + (w_l^x)^T v_l^x + (w_u^x)^T v_u^x - p & \leq & 0, \\ & & v_l^c, v_u^c, v_l^x, v_u^x \geq 0 & & \end{array} \tag{13.4}$$

minimizes the weighted sum of changes to the bounds that makes the problem feasible. The variables $(v_l^c)_i$, $(v_u^c)_i$, $(v_l^x)_i$ and $(v_u^c)_i$ are *elasticity* variables because they allow a constraint to be violated and hence add some elasticity to the problem. For instance, the elasticity variable $(v_l^c)_i$ shows how much the lower bound $(l^c)_i$ should be relaxed to make the problem feasible. Since p is minimized and

$$(w_l^c)^T v_l^c + (w_u^c)^T v_u^c + (w_l^x)^T v_l^x + (w_u^x)^T v_u^x - p \leq 0, \tag{13.5}$$

a large $(w_l^c)_i$ tends to imply that the elasticity variable $(v_l^c)_i$ will be small in an optimal solution.

The reader may want to verify that the problem (13.4) is always feasible given the assumptions (13.2) and (13.3).

Please note that if a weight is negative then the resulting problem (13.4) is unbounded.

The weights w_l^c, w_u^c, w_l^x, and w_u^x can be regarded as a costs (penalties) for violating the associated constraints. Thus a higher weight implies that higher priority is given to the satisfaction of the associated constraint.

The main idea can now be presented as follows. If you have an infeasible problem, then form the problem (13.4) and optimize it. Next inspect the optimal solution $(v_l^c)^*, (v_u^c)^*, (v_l^x)^*$, and $(v_u^x)^*$ to problem (13.4). This solution provides a suggested relaxation of the bounds that will make the problem feasible.

Assume that p^* is an optimal objective value to (13.4). An extension of the idea presented above is to solve the problem

$$\begin{array}{lcccl} \text{minimize} & & c^T x & & \\ \text{subject to} \quad l^c & \leq & Ax + v_l^c - v_u^c & \leq & u^c, \\ l^x & \leq & x + v_l^x - v_u^x & \leq & u^x, \\ & & (w_l^c)^T v_l^c + (w_u^c)^T v_u^c + (w_l^x)^T v_l^x + (w_u^x)^T v_u^x - p & \leq & 0, \\ & & p & = & p^*, \\ & & v_l^c, v_u^c, v_l^x, v_u^x \geq 0 & & \end{array} \tag{13.6}$$

which minimizes the true objective while making sure that total weighted violations of the bounds is minimal, i.e. equals to p^*.

13.2 Feasibility repair in MOSEK

MOSEK includes functionality that help you construct the problem (13.4) simply by passing a set of weights to MOSEK. This can be used for linear, quadratic, and conic optimization problems, possibly having integer constrained variables.

13.2.1 Usage of negative weights

As the problem (13.4) is presented it does not make sense to use negative weights since that makes the problem unbounded. Therefore, if the value of a weight is negative MOSEK fixes the associated elasticity variable to zero, e.g. if

$$(w_l^c)_i < 0$$

then MOSEK imposes the bound

$$(v_l^c)_i \leq 0.$$

This implies that the lower bound on the ith constraint will not be violated. (Clearly, this could also imply that the problem is infeasible so negative weight should be used with care). Associating a negative weights with a constraint tells MOSEK that the constraint should not be relaxed.

13.2.2 Automatical naming

MOSEK can automatically create a new problem of the form (13.4) starting from an existing problem by adding the elasticity variables and the extra constraints. Specificly, the variables v_l^c, v_u^c, v_l^x, v_u^x, and p are appended to existing variable vector x in their natural order. Moreover, the constraint (13.5) is appended to the constraints.

The new variables and constraints are automatically given names as follows:

- The names of the variables $(v_l^c)_i$ and $(v_u^c)_i$ are constructed from the name of the ith constraint. For instance, if the 9th original constraint is named `c9`, then by default $(v_l^c)_9$ and $(v_u^c)_9$ are given the names `LO*c9` and `UP*c9` respectively. If necessary, the character "`*`" can be replaced by a different string by changing the
  `MSK_SPAR_FEASREPAIR_NAME_SEPARATOR`
  parameter.

- The additional constraints

  $$l^x \leq x + v_l^x - v_u^x \leq u^x$$

  are given names as follows. There is exactly one constraint per variable in the original problem, and thus the ith of these constraints is named after the ith variable in the original problem. For instance, if the first original variable is named "`x0`", then the first of the above constraints is named "`MSK-x1`". If necessary, the prefix "`MSK-`" can be replaced by a different string by changing the
  `MSK_SPAR_FEASREPAIR_NAME_PREFIX`
  parameter.

- The variable p is by default given the name `WSUMVIOLVAR`, and the constraint (13.5) is given the name `WSUMVIOLCON`.

  The substring "`WSUMVIOL`" can be replaced by a different string by changing the
  `MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL`
  parameter.

13.2.3 An example

Consider the example linear optimization

$$\begin{array}{lccccc} \text{minimize} & -10x_1 & & -9x_2, & & \\ \text{subject to} & 7/10x_1 & + & 1x_2 & \leq & 630, \\ & 1/2x_1 & + & 5/6x_2 & \leq & 600, \\ & 1x_1 & + & 2/3x_2 & \leq & 708, \\ & 1/10x_1 & + & 1/4x_2 & \leq & 135, \\ & x_1, & & x_2 & \geq & 0. \\ & & x_2 \geq 650 & & & \end{array} \tag{13.7}$$

This is an infeasible problem. Now suppose we wish to use MOSEK to suggest a modification to the bounds that makes the problem feasible.

The command

```
mosek -d MSK_IPAR_FEASREPAIR_OPTIMIZE
MSK_FEASREPAIR_OPTIMIZE_PENALTY -d
MSK_IPAR_OPF_WRITE_SOLUTIONS MSK_ON  feasrepair.lp
-infrepo minv.opf
```

writes the problem (13.4) and it's solution to an OPF formatted file. In this case the file `minv.opf`.

The parameter

```
MSK_IPAR_FEASREPAIR_OPTIMIZE
```

controls whether the function returns the problem (13.4) or the problem (13.6). In the case

```
MSK_IPAR_FEASREPAIR_OPTIMIZE
```

is equal to

```
MSK_FEASREPAIR_OPTIMIZE_NONE
```

then (13.4) is returned, but the problem is not solved. For `MSK_FEASREPAIR_OPTIMIZE_PENALTY` the problem (13.4) is returned and solved. Finally for `MSK_FEASREPAIR_OPTIMIZE_COMBINED` (13.6) is returned and solved.

Chapter 14

Sensitivity analysis

14.1 Introduction

Given an optimization problem it is often useful to obtain information about how the optimal objective value change when the problem parameters are perturbed. For instance assume that a bound represents a capacity of a machine. Now it might be possible to expand the capacity for a certain cost and hence it worthwhile to know what the value of additional capacity is. This is precisely the type of questions sensitivity analysis deals with.

Analyzing how the optimal objective value changes when the problem data is changed is called sensitivity analysis.

14.2 Restrictions

Currently, sensitivity analysis is only available for continuous linear optimization problems. Moreover, MOSEK can only deal with perturbations in bounds or objective coefficients.

14.3 References

The book [13] discusses the classical sensitivity analysis in Chapter 10 whereas the book [20, Chapter 19] presents a modern introduction to sensitivity analysis. Finally, it is recommended to read the short paper [23] to avoid some of the pitfalls associated with sensitivity analysis.

14.4 Sensitivity analysis for linear problems

14.4.1 The optimal objective value function

Assume we are given the problem

$$\begin{array}{rclcccl} z(l^c, u^c, l^x, u^x, c) & = & \text{minimize} & & c^T x & & \\ & & \text{subject to} & l^c \leq & Ax & \leq & u^c, \\ & & & & l^x \leq x \leq u^x, & & \end{array} \tag{14.1}$$

and we want to know how the optimal objective value changes as l_i^c is perturbed. In order to answer this question then define the perturbed problem for l_i^c as follows

$$\begin{array}{rclccl} f_{l_i^c}(\beta) & = & \text{minimize} & & c^T x & \\ & & \text{subject to} & l^c + \beta e_i \leq & Ax & \leq u^c, \\ & & & & l^x \leq x \leq u^x, & \end{array} \tag{14.2}$$

where e_i is the ith column of the identity matrix. The function

$$f_{l_i^c}(\beta) \tag{14.3}$$

shows the optimal objective value as a function of β. Note a change in β corresponds to a perturbation in l_i^c and hence (14.3) shows the optimal objective value as a function of l_i^c.

It is possible to prove that the function (14.3) is a piecewise linear and convex function i.e. the function may look like the illustration in Figure 14.1.

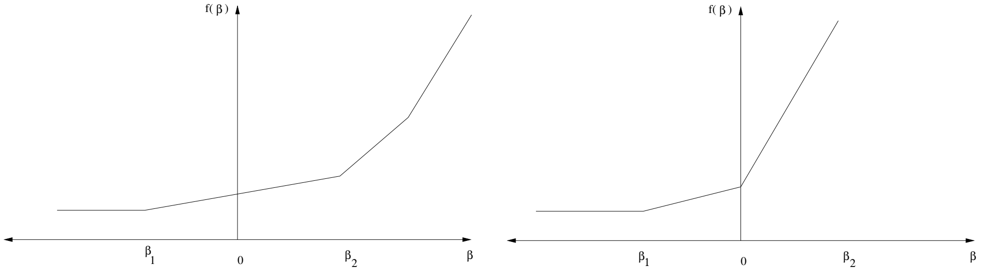

Figure 14.1: The optimal value function $f_{l_i^c}(\beta)$. Left: $\beta = 0$ is in the interior of linearity interval. Right: $\beta = 0$ is a breakpoint.

Clearly, if the function $f_{l_i^c}(\beta)$ does not change much when β is changed, then we can conclude that the optimal objective value is insensitive to changes in l_i^c. Therefore, we are interested in how $f_{l_i^c}(\beta)$ changes for small changes in β. Now define

$$f'_{l_i^c}(0) \tag{14.4}$$

to be the so called *shadow price* related to l_i^c. The shadow price specifies how the objective value changes for small changes in β around zero. Moreover, we are interested in the so called *linearity interval*

$$\beta \in [\beta_1, \beta_2] \tag{14.5}$$

for which

$$f'_{l_i^c}(\beta) = f'_{l_i^c}(0). \tag{14.6}$$

To summarize the sensitivity analysis provides a shadow price and the linearity interval in which the shadow price is constant.

The reader may have noticed that we are sloppy in the definition of the shadow price. The reason is that the shadow price is not defined in the right example in Figure 14.1 because the function $f_{l_i^c}(\beta)$ is not differentiable for $\beta = 0$. However, in that case we can define a left and a right shadow price and a left and a right linearity interval.

In the above discussion we only discussed changes in l_i^c. We define the other optimal objective value functions as follows

$$\begin{array}{rcll} f_{u_i^c}(\beta) & = & z(l^c, u^c + \beta e_i, l^x, u^x, c), & i = 1, \ldots, m, \\ f_{l_j^x}(\beta) & = & z(l^c, u^c, l^x + \beta e_j, u^x, c), & j = 1, \ldots, n, \\ f_{u_j^x}(\beta) & = & z(l^c, u^c, l^x, u^x + \beta e_j, c), & j = 1, \ldots, n, \\ f_{c_j}(\beta) & = & z(l^c, u^c, l^x, u^x, c + \beta e_j), & j = 1, \ldots, n. \end{array} \tag{14.7}$$

Given these definitions it should be clear how linearity intervals and shadow prices are defined for the parameters u_i^c etc.

14.4.1.1 Equality constraints

In MOSEK a constraint can be specified as either an equality constraints or a ranged constraints. Suppose constraint i is an equality constraint. We then define the optimal value function for constraint i by

$$f_{e_i^c}(\beta) = z(l^c + \beta e_i, u^c + \beta e_i, l^x, u^x, c) \tag{14.8}$$

Thus for a equality constraint the upper and lower bound (which are equal) are perturbed simultaneously. From the point of view of MOSEK sensitivity analysis a ranged constrain with $l_i^c = u_i^c$ therefore differs from an equality constraint.

14.4.2 The basis type sensitivity analysis

The classical sensitivity analysis discussed in most textbooks about linear optimization, e.g. [13, Chapter 10], is based on an optimal basic solution or equivalently on an optimal basis. This method may produce misleading results [20, Chapter 19] but is **computationally cheap**. Therefore, and for historical reasons this method is available in MOSEK.

We will now briefly discuss the basis type sensitivity analysis. Given an optimal basic solution which provides a partition of variables into basic and non-basic variables then the basis type sensitivity analysis computes the linearity interval $[\beta_1, \beta_2]$ such that the basis remains optimal for the perturbed problem. A shadow price associated with the linearity interval is also computed. However, it is well known that an optimal basic solution may not be unique and therefore the result depends on the optimal basic solution employed in the sensitivity analysis. This implies the computed interval is only a subset of the largest interval for which the shadow price is constant. Furthermore, the optimal objective value function might have a breakpoint for $\beta = 0$. In this case the basis type sensitivity method will only provide a subset of either the left or the right linearity interval.

In summary the basis type sensitivity analysis is computationally cheap but does not provide complete information. Hence, the results of the basis type sensitivity analysis should be used with care.

14.4.3 The optimal partition type sensitivity analysis

Another method for computing the complete linearity interval is called the *optimal partition type sensitivity analysis*. The main drawback to the optimal partition type sensitivity analysis is it is computationally expensive. This type of sensitivity analysis is currently provided as an experimental feature in MOSEK.

Given optimal primal and dual solutions to (14.1) i.e. x^* and $((s_l^c)^*, (s_u^c)^*, (s_l^x)^*, (s_u^x)^*)$ then the optimal objective value is given by

$$z^* := c^T x^*. \tag{14.9}$$

The left and right shadow prices σ_1 and σ_2 for l_i^c is given by the pair of optimization problems

$$\begin{array}{rcll} \sigma_1 & = & \text{minimize} & \begin{array}{ccc} & e_i^T s_l^c & \\ A^T(s_l^c - s_u^c) + s_l^x - s_u^x & = & c, \\ (l_c)^T(s_l^c) - (u_c)^T(s_u^c) + (l_x)^T(s_l^x) - (u_x)^T(s_u^x) & = & z^*, \\ & s_l^c, s_u^c, s_l^c, s_u^x \geq 0 & \end{array} \\ & & \text{subject to} & \end{array} \tag{14.10}$$

and

$$\begin{array}{rcll} \sigma_2 & = & \text{maximize} & \begin{array}{ccc} & e_i^T s_l^c & \\ A^T(s_l^c - s_u^c) + s_l^x - s_u^x & = & c, \\ (l_c)^T(s_l^c) - (u_c)^T(s_u^c) + (l_x)^T(s_l^x) - (u_x)^T(s_u^x) & = & z^*, \\ & s_l^c, s_u^c, s_l^c, s_u^x \geq 0. & \end{array} \\ & & \text{subject to} & \end{array} \tag{14.11}$$

The above two optimization problems makes it easy to interpret-ate the shadow price. Indeed assume that $((s_l^c)^*, (s_u^c)^*, (s_l^x)^*, (s_u^x)^*)$ is an arbitrary optimal solution then it must hold

$$(s_l^c)_i^* \in [\sigma_1, \sigma_2]. \tag{14.12}$$

Next the linearity interval $[\beta_1, \beta_2]$ for l_i^c is computed by solving the two optimization problems

$$\begin{array}{rcllcccl} \beta_1 & = & \text{minimize} & & & \beta & & \\ & & \text{subject to} & l^c + \beta e_i & \leq & Ax & \leq & u^c, \\ & & & & & c^T x - \sigma_1 \beta & = & z^*, \\ & & & & & l^x \leq x \leq u^x, & & \end{array} \tag{14.13}$$

and

$$\begin{array}{rcllcccl} \beta_2 & = & \text{maximize} & & & \beta & & \\ & & \text{subject to} & l^c + \beta e_i & \leq & Ax & \leq & u^c, \\ & & & & & c^T x - \sigma_2 \beta & = & z^*, \\ & & & & & l^x \leq x \leq u^x. & & \end{array} \tag{14.14}$$

The linearity intervals and shadow prices for u_i^c, l_j^x, and u_j^x can be computed in a similar way to how it is computed for l_i^c.

The left and right shadow price for c_j denoted σ_1 and σ_2 respectively is given by the pair optimization problems

$$\begin{array}{rcllcccl} \sigma_1 & = & \text{minimize} & & & e_j^T x & & \\ & & \text{subject to} & l^c + \beta e_i & \leq & Ax & \leq & u^c, \\ & & & & & c^T x & = & z^*, \\ & & & & & l^x \leq x \leq & u^x & \end{array} \tag{14.15}$$

and

$$\begin{array}{rcllcccl} \sigma_2 & = & \text{maximize} & & & e_j^T x & & \\ & & \text{subject to} & l^c + \beta e_i & \leq & Ax & \leq & u^c, \\ & & & & & c^T x & = & z^*, \\ & & & & & l^x \leq x \leq & u^x. & \end{array} \tag{14.16}$$

Once again the above two optimization problems makes it easy to interpret-ate the shadow prices. Indeed assume that x^* is an arbitrary primal optimal solution then it must hold

$$x_j^* \in [\sigma_1, \sigma_2]. \tag{14.17}$$

The linearity interval $[\beta_1, \beta_2]$ for a c_j is computed as follows

$$\begin{array}{rclccl} \beta_1 & = & \text{minimize} & \beta & & \\ & & \text{subject to} & A^T(s_l^c - s_u^c) + s_l^x - s_u^x & = & c + \beta e_j, \\ & & & (l_c)^T(s_l^c) - (u_c)^T(s_u^c) + (l_x)^T(s_l^x) - (u_x)^T(s_u^x) - \sigma_1 \beta & \leq & z^*, \\ & & & s_l^c, s_u^c, s_l^c, s_u^x \geq 0 & & \end{array} \tag{14.18}$$

and

$$\begin{array}{rclccl} \beta_2 & = & \text{maximize} & \beta & & \\ & & \text{subject to} & A^T(s_l^c - s_u^c) + s_l^x - s_u^x & = & c + \beta e_j, \\ & & & (l_c)^T(s_l^c) - (u_c)^T(s_u^c) + (l_x)^T(s_l^x) - (u_x)^T(s_u^x) - \sigma_2 \beta & \leq & z^*, \\ & & & s_l^c, s_u^c, s_l^c, s_u^x \geq 0. & & \end{array} \tag{14.19}$$

14.4.4 An example

As an example we will use the following transportation problem. Consider the problem of minimizing the transportation cost between a number of production plants and stores. Each plant supplies a number of goods and each store has a given demand that must be met. Supply, demand and cost of transportation per unit are shown in Figure 14.2.

If we denote the number of transported goods from location i to location j by x_{ij}, the problem can be formulated as the linear optimization problem

minimize

$$1x_{11} + 2x_{12} + 5x_{23} + 2x_{24} + 1x_{31} + 2x_{33} + 1x_{34} \tag{14.20}$$

subject to

$$\begin{array}{ccccccccccccccl} x_{11} & + & x_{12} & & & & & & & & & & & \leq & 400, \\ & & & & x_{23} & + & x_{24} & & & & & & & \leq & 1200, \\ & & & & & & & & x_{31} & + & x_{33} & + & x_{34} & \leq & 1000, \\ x_{11} & & & & & & & + & x_{31} & & & & & = & 800, \\ & & x_{12} & & & & & & & & & & & = & 100, \\ & & & & x_{23} & + & & & & & x_{33} & & & = & 500, \\ & & & & & & x_{24} & + & & & & & x_{34} & = & 500, \\ x_{11}, & & x_{12}, & & x_{23}, & & x_{24}, & & x_{31}, & & x_{33}, & & x_{34} & \geq & 0. \end{array} \tag{14.21}$$

The basis type and the optimal partition type sensitivity results for the transportation problem is shown in Table 14.1 and 14.2 respectively.

Looking at the results from the optimal partition type sensitivity analysis we see that for the constraint number 1 we have $\sigma_1 \neq \sigma_2$ and $\beta_1 \neq \beta_2$. Therefore, we have a left linearity interval of $[-300, 0]$ and a right interval of $[0, 500]$. The corresponding left and right shadow price is 3 and 1 respectively. This implies if the upper bound on constraint 1 increases by

$$\beta \in [0, \beta_1] = [0, 500] \tag{14.22}$$

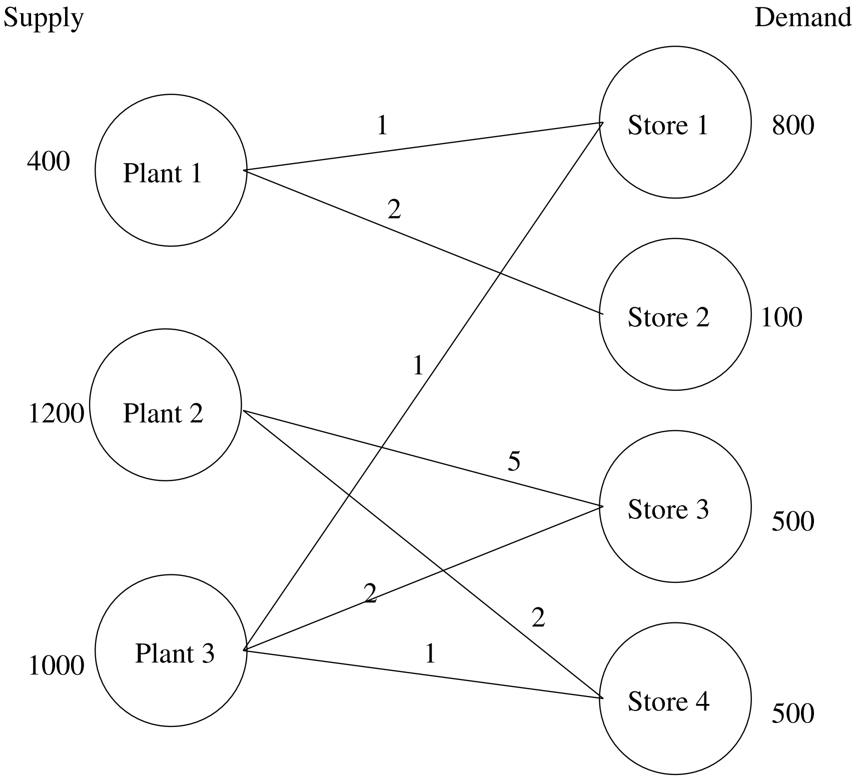

Figure 14.2: Supply, demand and cost of transportation.

Basis type

Con.	β_1	β_2	σ_1	σ_2
1	-300.00	0.00	3.00	3.00
2	-700.00	$+\infty$	0.00	0.00
3	-500.00	0.00	3.00	3.00
4	-0.00	500.00	4.00	4.00
5	-0.00	300.00	5.00	5.00
6	-0.00	700.00	5.00	5.00
7	-500.00	700.00	2.00	2.00
Var.	β_1	β_2	σ_1	σ_2
x_{11}	$-\infty$	300.00	0.00	0.00
x_{12}	$-\infty$	100.00	0.00	0.00
x_{23}	$-\infty$	0.00	0.00	0.00
x_{24}	$-\infty$	500.00	0.00	0.00
x_{31}	$-\infty$	500.00	0.00	0.00
x_{33}	$-\infty$	500.00	0.00	0.00
x_{34}	-0.000000	500.00	2.00	2.00

Optimal partition type

Con.	β_1	β_2	σ_1	σ_2
1	-300.00	500.00	3.00	1.00
2	-700.00	$+\infty$	-0.00	-0.00
3	-500.00	500.00	3.00	1.00
4	-500.00	500.00	2.00	4.00
5	-100.00	300.00	3.00	5.00
6	-500.00	700.00	3.00	5.00
7	-500.00	700.00	2.00	2.00
Var.	β_1	β_2	σ_1	σ_2
x_{11}	$-\infty$	300.00	0.00	0.00
x_{12}	$-\infty$	100.00	0.00	0.00
x_{23}	$-\infty$	500.00	0.00	2.00
x_{24}	$-\infty$	500.00	0.00	0.00
x_{31}	$-\infty$	500.00	0.00	0.00
x_{33}	$-\infty$	500.00	0.00	0.00
x_{34}	$-\infty$	500.00	0.00	2.00

Table 14.1: Ranges and shadow prices related to bounds on constraints and variables. Left: Results for basis type sensitivity analysis. Right: Results for the optimal partition type sensitivity analysis.

Basis type

Var.	β_1	β_2	σ_1	σ_2
c_1	$-\infty$	3.00	300.00	300.00
c_2	$-\infty$	∞	100.00	100.00
c_3	-2.00	∞	0.00	0.00
c_4	$-\infty$	2.00	500.00	500.00
c_5	-3.00	∞	500.00	500.00
c_6	$-\infty$	2.00	500.00	500.00
c_7	-2.00	∞	0.00	0.00

Optimal partition type

Var.	β_1	β_2	σ_1	σ_2
c_1	$-\infty$	3.00	300.00	300.00
c_2	$-\infty$	∞	100.00	100.00
c_3	-2.00	∞	0.00	0.00
c_4	$-\infty$	2.00	500.00	500.00
c_5	-3.00	∞	500.00	500.00
c_6	$-\infty$	2.00	500.00	500.00
c_7	-2.00	∞	0.00	0.00

Table 14.2: Ranges and shadow prices related to the objective coefficients. Left: Results for basis type sensitivity analysis. Right: Results for the optimal partition type sensitivity analysis.

then the optimal objective value will decrease by the value

$$\sigma_2 \beta = 1\beta. \tag{14.23}$$

Correspondingly, if the upper bound on constraint 1 is decreased by

$$\beta \in [0, 300] \tag{14.24}$$

then the optimal objective value will increased by the value

$$\sigma_1 \beta = 3\beta. \tag{14.25}$$

14.5 Sensitivity analysis with the command line tool

A sensitivity analysis can be performed with the MOSEK command line tool using the command

```
mosek myproblem.mps -sen sensitivity.ssp
```

where `sensitivity.ssp` is a file in the format described in the next section. The `ssp` file describes which parts of the problem the sensitivity analysis should be performed on.

By default results are written to a file named `myproblem.sen`. If necessary, this filename can be changed by setting the
`MSK_SPAR_SENSITIVITY_RES_FILE_NAME`
parameter By default a basis type sensitivity analysis is performed. However, the type of sensitivity analysis (basis or optimal partition) can be changed by setting the parameter
`MSK_IPAR_SENSITIVITY_TYPE`
appropriately. Following values are accepted for this parameter:

- `MSK_SENSITIVITY_TYPE_BASIS`
- `MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION`

It is also possible to use the command line

```
mosek myproblem.mps -d MSK_IPAR_SENSITIVITY_ALL MSK_ON
```

in which case a sensitivity analysis on all the parameters is performed.

14.5.1 Sensitivity analysis specification file

MOSEK employs an MPS like file format to specify on which model parameters the sensitivity analysis should be performed. As the optimal partition type sensitivity analysis can be computationally expensive it is important to limit the sensitivity analysis.

The format of the sensitivity specification file is shown in figure 14.3, where capitalized names are keywords, and names in brackets are names of the constraints and variables to be included in the analysis.

The sensitivity specification file has three sections, i.e.

- `BOUNDS CONSTRAINTS`: Specifies on which bounds on constraints the sensitivity analysis should be performed.

```
* A comment
BOUNDS CONSTRAINTS
 U|L|UL [cname1]
 U|L|UL [cname2]-[cname3]
BOUNDS VARIABLES
 U|L|UL [vname1]
 U|L|UL [vname2]-[vname3]
OBJECTIVE VARIABLES
 [vname1]
 [vname2]-[vname3]
```

Figure 14.3: The sensitivity analysis file format.

- `BOUNDS VARIABLES`: Specifies on which bounds on variables the sensitivity analysis should be performed.
- `OBJECTIVE VARIABLES`: Specifies on which objective coefficients the sensitivity analysis should be performed.

A line in the body of a section must begin with a whitespace. In the `BOUNDS` sections one of the keys `L`, `U`, and `LU` must appear next. These keys specify whether the sensitivity analysis is performed on the lower bound, on the upper bound, or on both the lower and the upper bound respectively. Next, a single constraint (variable) or range of constraints (variables) is specified.

Recall from Section 14.4.1.1 that equality constraints are handled in a special way. Sensitivity analysis of an equality constraint can be specified with either `L`, `U`, or `LU`, all indicating the same, namely that upper and lower bounds (which are equal) are perturbed simultaneously.

As an example consider

```
BOUNDS CONSTRAINTS
 L  "cons1"
 U  "cons2"
 LU "cons3"-"cons6"
```

which requests that sensitivity analysis is performed on the lower bound of the constraint named `cons1`, on the upper bound of the constraint named `cons2`, and on both lower and upper bound on the constraints named `cons3` to `cons6`.

It is allowed to use indexes instead of names, for instance

```
BOUNDS CONSTRAINTS
 L  "cons1"
 U  2
 LU 3 - 6
```

The character “*” indicates that the line contains a comment and is ignored.

14.5.2 Example: Sensitivity analysis from command line

As an example consider the `sensitivity.ssp` file shown in Figure 14.4.

The command

```
* Comment 1

BOUNDS CONSTRAINTS
 U "c1"         * Analyze upper bound for constraint named c1
 U 2            * Analyze upper bound for the second constraint
 U 3-5          * Analyze upper bound for constraint number 3 to number 5

BOUNDS VARIABLES
 L 2-4          * This section specifies which bounds on variables should be analyzed
 L "x11"
OBJECTIVE VARIABLES
 "x11"          * This section specifies which objective coefficients should be analyzed
 2
```

Figure 14.4: Example of the sensitivity file format.

```
mosek transport.lp -sen sensitivity.ssp  -d MSK_IPAR_SENSITIVITY_TYPE MSK_SENSITIVITY_TYPE_BASIS
```

produces the `transport.sen` file shown below.

```
BOUNDS CONSTRAINTS
INDEX    NAME             BOUND    LEFTRANGE        RIGHTRANGE       LEFTPRICE        RIGHTPRICE
0        c1               UP       -6.574875e-18    5.000000e+02     1.000000e+00     1.000000e+00
2        c3               UP       -6.574875e-18    5.000000e+02     1.000000e+00     1.000000e+00
3        c4               FIX      -5.000000e+02    6.574875e-18     2.000000e+00     2.000000e+00
4        c5               FIX      -1.000000e+02    6.574875e-18     3.000000e+00     3.000000e+00
5        c6               FIX      -5.000000e+02    6.574875e-18     3.000000e+00     3.000000e+00

BOUNDS VARIABLES
INDEX    NAME             BOUND    LEFTRANGE        RIGHTRANGE       LEFTPRICE        RIGHTPRICE
2        x23              LO       -6.574875e-18    5.000000e+02     2.000000e+00     2.000000e+00
3        x24              LO       -inf             5.000000e+02     0.000000e+00     0.000000e+00
4        x31              LO       -inf             5.000000e+02     0.000000e+00     0.000000e+00
0        x11              LO       -inf             3.000000e+02     0.000000e+00     0.000000e+00

OBJECTIVE VARIABLES
INDEX    NAME                      LEFTRANGE        RIGHTRANGE       LEFTPRICE        RIGHTPRICE
0        x11                       -inf             1.000000e+00     3.000000e+02     3.000000e+02
2        x23                       -2.000000e+00    +inf             0.000000e+00     0.000000e+00
```

14.5.3 Controlling log output

Setting the parameter

`MSK_IPAR_LOG_SENSITIVITY`

to 1 or 0 (default) controls whether or not the results from sensitivity calculations are printed to the message stream.

The parameter

`MSK_IPAR_LOG_SENSITIVITY_OPT`

controls the amount of debug information on internal calculations from the sensitivity analysis.

Appendix A

MOSEK command line tool reference

A.1 Introduction

The MOSEK command line tool is used to solve optimization problems from the operating system command line. It is invoked as follows

```
mosek [options] [filename]
```

where both `[options]` and `[filename]` are optional arguments. `[filename]` is a file describing the optimization problems and is either a MPS file or AMPL nl file. `[options]` consists of command line arguments that modifies the behavior of MOSEK.

A.2 Command line arguments

The following list shows the possible command-line arguments for MOSEK:

`-a`	MOSEK runs in AMPL mode.
`-AMPL`	The input file is an AMPL nl file.
`-basi name`	Input basis solution file `name`.
`-baso name`	Output basis solution file `name`.
`-brni name`	`name` is the filename of a variable branch order file to be read.
`-brno name`	`name` is the filename of a variable branch order file to be written.
`-d name val`	Assigns the value `val` to the parameter named `name`.
`-dbgmem name`	Name of memory debug file. Write memory debug information to file `name`.
`-f`	Complete license information is printed.

`-h` Prints out help information for MOSEK.

`-inti name` Input integer solution file `name`.

`-into name` Output integer solution file `name`.

`-itri name` Input interior point solution file `name`.

`-itro name` Output interior point solution file `name`.

`-info name` Infeasible subproblem output file `name`.

`-infrepo name` Feasibility reparation output file

`-pari name` Input parameter file `name`. Equivalent to `-p`.

`-paro name` Output parameter file `name`.

`-L name` `name` of the license file.

`-l name` `name` of the license file.

`-max` Forces MOSEK to maximize the objective.

`-min` Forces MOSEK to minimize the objective.

`-n` Ignore errors in subsequent paramter settings.

`-p name` New parameter settings are read from a file named `name`.

`-q name` Name of a optional log file.

`-r` If the option is present, the program returns -1 if an error ocurred otherwise 0.

`-rout name` If the option is present, the program writes the return code to file 'name'.

`-sen file` Perform sensitivity analysis based on file.

`-silent` As little information as possible is send to the terminal.

`-v` The MOSEK version number is printed and no optimization is performed.

`-w` If this options is included, then MOSEK will wait for a license.

`-=` Lists the parameter database.

`-?` Same as the -h option.

A.3 The parameter file

Occasionally system or algorithmic parameters in MOSEK should be changed be the user. One way of the changing parameters is to use a so-called parameter file which is a plain text file. It can for example can have the format

```
BEGIN MOSEK
% This is a comment.
% The subsequent line tells MOSEK that an optimal
% basis should be computed by the interior-point optimizer.
MSK_IPAR_INTPNT_BASIS       MSK_BI_ALWAYS
MSK_DPAR_INTPNT_TOL_PFEAS   1.0e-9
END MOSEK
```

Note that the file begins with an `BEGIN MOSEK` and is terminated with an `END MOSEK`, this is required. Moreover, everything that appears after an `%` is considered to be a comment and is ignored. Similarly, empty lines are ignored. The important lines are those which begins with a valid MOSEK parameter name such as `MSK_IPAR_INTPNT_BASIS`. Immediately after parameter name follows the new value for the parameter. All the MOSEK parameter names are listed in Appendix H.

A.3.1 Using the parameter file

The parameter file can be given any name, but let us assume it has the name `mosek.par`. If MOSEK should use the parameter settings in that file, then `-p mosek.par` should be on the command line when MOSEK is invoked. An example of such a command line is

```
mosek -p mosek.par afiro.mps
```

Appendix B

The MPS file format

MOSEK supports the standard MPS format with some extensions. For a detailed description of the MPS format the book by Nazareth [19] is a good reference.

B.1 The MPS file format

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

$$\begin{array}{rcccl} l^c & \leq & Ax + q(x) & \leq & u^c, \\ l^x & \leq & x & \leq & u^x, \\ & & x \in \mathcal{C}, & & \\ & & x_{\mathcal{J}} \text{ integer}, & & \end{array} \tag{B.1}$$

where

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

  $$q_i(x) = 1/2 x^T Q^i x$$

  where it is assumed that

  $$Q^i = (Q^i)^T. \tag{B.2}$$

  Please note the explicit 1/2 in the quadratic term and that Q^i is required to be symmetric.
- $\mathcal{C}$ is a convex cone.

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

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

```
*        1         2         3         4         5         6
*23456789012345678901234567890123456789012345678901234567890
NAME          [name]
OBJSENSE
    [objsense]
OBJNAME
    [objname]
ROWS
 ?  [cname1]
COLUMNS
    [vname1]  [cname1]    [value1]     [vname3]  [value2]
RHS
    [name]    [cname1]    [value1]     [cname2]  [value2]
RANGES
    [name]    [cname1]    [value1]     [cname2]  [value2]
QSECTION      [cname1]
    [vname1]  [vname2]    [value1]     [vname3]  [value2]
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 MOSEK specific extensions of 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 Section B.5 for details.

B.1.1 An example

A concrete example of a MPS file is presented below:

```
NAME          EXAMPLE
OBJSENSE
    MIN
ROWS
 N  obj
 L  c1
 L  c2
 L  c3
 L  c4
COLUMNS
    x1        obj       -10.0          c1        0.7
    x1        c2        0.5            c3        1.0
    x1        c4        0.1
    x2        obj       -9.0           c1        1.0
    x2        c2        0.8333333333   c3        0.66666667
    x2        c4        0.25
RHS
    rhs       c1        630.0          c2        600.0
    rhs       c3        708.0          c4        135.0
ENDATA
```

Subsequently each individual section in the MPS format is discussed.

B.1.2 `NAME`

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

B.1.3 `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.

B.1.4 `OBJNAME` (optional)

This is an optional section that can be used to specify the name of the row that is used as objective function. The `OBJNAME` section contains one line at most which has the form

```
objname
```

`objname` should be a valid row name.

B.1.5 `ROWS`

A record in the `ROWS` section has the form

```
? [cname1]
```

where the requirements for the fields are as follows:

Field	Starting position	Maximum width	Required	Description
?	2	1	Yes	Constraint key
`[cname1]`	5	8	Yes	Constraint name

Hence, in this section each constraint is assigned an 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 the values `E`, `G`, `L`, or `N` whith ther following interpretation:

Constraint type	l_i^c	u_i^c
`E`	finite	l_i^c
`G`	finite	∞
`L`	$-\infty$	finite
`N`	$-\infty$	∞

In the MPS format an 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.

B.1.6 `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	Maximum 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.

B.1.7 `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	Maximum width	Re- quired	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 ith constraint and v_1 denotes the value specified by `[value1]`, then the interpretation of v_1 is:

Constraint type	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.

B.1.8 `RANGES` (optional)

A record in this section has the form

```
[name]    [cname1]    [value1]     [cname2]  [value2]
```

where the requirements for each fields are as follows:

Field	Starting position	Maximum width	Required	Description
`[name]`	5	8	Yes	Name of the `RANGE` vector
`[cname1]`	15	8	Yes	Constraint name
`[value1]`	25	12	Yes	Numerical value
`[cname2]`	40	8	No	Constraint name
`[value2]`	50	12	No	Numerical value

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

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

B.1.9 `QSECTION` (optional)

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

```
    [vname1]  [vname2]   [value1]    [vname3]  [value2]
```

where the requirements for each field are:

Field	Starting position	Maximum width	Required	Description
`[vname1]`	5	8	Yes	Variable name
`[vname2]`	15	8	Yes	Variable name
`[value1]`	25	12	Yes	Numerical value
`[vname3]`	40	8	No	Variable name
`[value2]`	50	12	No	Numerical value

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

The example

$$\begin{array}{lccc} \text{minimize} & -x_2 + 0.5(2x_1^2 - 2x_1x_3 + 0.2x_2^2 + 2x_3^2) & & \\ \text{subject to} & x_1 + x_2 + x_3 & \geq & 1, \\ & x \geq 0 & & \end{array}$$

has the following MPS file representation

```
NAME          qoexp
ROWS
 N  obj
 G  c1
COLUMNS
    x1        c1        1
    x2        obj       -1
    x2        c1        1
    x3        c1        1
RHS
    rhs       c1        1
QSECTION      obj
    x1        x1        2
    x1        x3        -1
    x2        x2        0.2
    x3        x3        2
ENDATA
```

Regarding the `QSECTION`s please note that:

- Only one `QSECTION` is allowed for each constraint.
- The `QSECTION`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.
- All entries specified in a `QSECTION` are assumed to belong to the lower triangular part of the quadratic term of Q.

B.1.10 `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	Maximum 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	Variable name

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 which 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	$\lceil v_1 \rceil$	∞	Yes
UI	unchanged	$\lfloor v_1 \rfloor$	Yes

v_1 is the value specified by `[value1]`.

B.1.11 `CSECTION` (optional)

The purpose of the `CSECTION` is to specify the constraint

$$x \in \mathcal{C}.$$

in (B.1).

It is assumed that $\mathcal{C}$ satisfies the following requirements. Let

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

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

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

Next define

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

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

- R set:

$$\mathcal{C}_t = \{x \in R^{n^t}\}.$$

- Quadratic cone:

$$\mathcal{C}_t = \left\{ x \in R^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\}. \tag{B.3}$$

- Rotated quadratic cone:

$$\mathcal{C}_t = \left\{ x \in R^{n^t} : 2x_1 x_2 \geq \sum_{j=3}^{n^t} x_j^2, \ x_1, x_2 \geq 0 \right\}. \tag{B.4}$$

In general, only quadratic and rotated quadratic cones are specified in the MPS file whereas membership of the R set is not. If a variable is not a member of any other cone then it is assumed to be a member of an R cone.

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

$$x_4 \geq \sqrt{x_5^2 + x_8^2} \tag{B.5}$$

and the rotated quadratic cone

$$2x_3x_7 \geq x_1^2 + x_8^2, \ x_3, x_7 \geq 0, \tag{B.6}$$

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        0.0          QUAD
    x4
    x5
    x8
CSECTION      koneb        0.0          RQUAD
    x7
    x3
    x1
    x0
```

This first `CSECTION` specifies the cone (B.5) which is given the name `konea`. This is a quadratic cone which is specified by the keyword `QUAD` in the `CSECTION` header. The 0.0 value in the `CSECTION` header is not used by the `QUAD` cone.

The second `CSECTION` specifies the rotated quadratic cone (B.6). Please note the keyword `RQUAD` in the `CSECTION` which is used to specify that the cone is a rotated quadratic cone instead of a quadratic cone. The 0.0 value in the `CSECTION` header is not used by the `RQUAD` cone.

In general, a `CSECTION` header has the format

```
CSECTION      [kname1]     [value1]     [ktype]
```

where the requirement for each field are as follows:

Field	Starting position	Maximum width	Required	Description
`[kname1]`	5	8	Yes	Name of the cone
`[value1]`	15	12	No	Cone parameter
`[ktype]`	25		Yes	Type of the cone.

The possible cone type keys are:

Cone type key	Members	Interpretation.
`QUAD`	≥ 1	Quadratic cone i.e. (B.3).
`RQUAD`	≥ 2	Rotated quadratic cone i.e. (B.4).

Please note that a quadratic cone must have at least one member whereas a rotated quadratic cone must have at least two members. A record in the `CSECTION` has the format

```
[vname1]
```

where the requirements for each field are

Field	Starting position	Maximum width	Re-quired	Description
`[vname1]`	2	8	Yes	A valid variable name

The most important restriction with respect to the `CSECTION` is that a variable must occur in only one `CSECTION`.

B.1.12 `ENDATA`

This keyword denotes the end of the MPS file.

B.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 compability 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

- IMPORTANT: 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.

B.3 General limitations

- An MPS file should be an ASCII file.

B.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.

B.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, it also presents two main limitations:

- By default a line in the MPS file must not contain more than 1024 characters. However, by modifying the parameter `MSK_IPAR_READ_MPS_WIDTH` an arbitrary large line width will be accepted.
- A name must not contain any blanks.

To use the free MPS format instead of the default MPS format the MOSEK parameter `MSK_IPAR_READ_MPS_FORMAT` should be changed.

Appendix C

The LP file format

MOSEK supports the LP file format with some extensions i.e. MOSEK can read and write LP formatted files.

C.1 A warning

The LP format is not a well-defined standard and hence different optimization packages may interpretate a specific LP formatted file differently.

C.2 The LP file format

The LP file format can specify problems on the form

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

where

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

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

  and it is assumed that

  $$Q^o = (Q^o)^T. \tag{C.1}$$
- $A \in R^{m \times n}$ is the constraint matrix.
- $l^c \in R^m$ is the lower limit on the activity for the constraints.

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

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

  where it is assumed that

  $$Q^i = (Q^i)^T. \tag{C.2}$$

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

C.2.1 The sections

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

C.2.1.1 The objective

The first section beginning with one of the keywords

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

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

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

The objective may be given a name by writing

```
myname:
```

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

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

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

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

```
x1 ^ 2
```

and

```
x1 * x2
```

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

An example of an objective section is:

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

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

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

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

C.2.1.2 The constraints

The second section beginning with one of the keywords

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

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

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

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

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

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

$$-5 \leq x_1 + x_2 \leq 5 \tag{C.3}$$

may be written as

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

By default MOSEK writes ranged constraints this way.

If the files must adhere to the LP standard, ranged constraints must either be split into upper bounded and lower bounded constraints or be written as en equality with a slack variable. For example the expression (C.3) may be written as

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

C.2.1.3 Bounds

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

```
bound
bounds
```

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

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

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

C.2.1.4 Variable types

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

```
bin
binaries
binary
```

and

```
gen
general
```

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

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

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

C.2.1.5 Terminating section

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

```
end
```

C.2.1.6 An example

A simple example of an LP file with two variables, four constraints and one integer variable is:

```
minimize
  -10 x1 -9 x2
subject to
  0.7 x1 +       x2 <= 630
  0.5 x1 + 0.833 x2 <= 600
      x1 + 0.667 x2 <= 708
  0.1 x1 + 0.025 x2 <= 135
bounds
  10 <= x1
  x1 <= +inf
  20 <= x2 <= 500
general
  x1
end
```

C.2.2 LP format peculiarities

C.2.2.1 Comments

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

C.2.2.2 Names

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

`!"#$%&()/,.;?@_'‘{}|~`

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

It is strongly recommended not to use double quotes (") in names.

C.2.2.3 Variable bounds

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

C.2.2.4 MOSEK specific extensions to the LP format

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

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

If an LP formatted file created by MOSEK should satisfies the strict definition, then the parameter

`MSK_IPAR_WRITE_LP_STRICT_FORMAT`

should be set; note, however, that some problems cannot be written correctly as a strict LP formatted file. For instance, all names are truncated to 16 characters and hence they may loose their uniqueness and change the problem.

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

Internally in MOSEK names may contain any (printable) character, many of which cannot be used in LP names. Setting the parameters

`MSK_IPAR_READ_LP_QUOTED_NAMES`

and

`MSK_IPAR_WRITE_LP_QUOTED_NAMES`

allows MOSEK to use quoted names. The first parameter tells MOSEK to remove quotes from quoted names e.g, `"x1"`, when reading LP formatted files. The second parameter tells MOSEK to put quotes around any semi-illegal name (names beginning with a number or a period) and fully illegal name (containing illegal characters). As double quote is a legal character in the LP format, quoting semi-illegal names makes them legal in the pure LP format as long as they are still shorter than 16 characters. Fully illegal names are still illegal in a pure LP file.

C.2.3 The strict LP format

The LP format is not a formal standard and different vendors have slightly different interpretations of the LP format. To make MOSEK's definition of the LP format more compatible whith the definitions of other vendors use the paramter setting

```
MSK_IPAR_WRITE_LP_STRICT_FORMAT MSK_ON
```

This setting may lead to truncation of some names and hence to an invalid LP file. The simple solution to this problem is to use the paramter setting

```
MSK_IPAR_WRITE_GENERIC_NAMES MSK_ON
```

which will cause all names to be renamed systematically in the output file.

C.2.4 Formatting of an LP file

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

```
MSK_IPAR_WRITE_LP_LINE_WIDTH
MSK_IPAR_WRITE_LP_TERMS_PER_LINE
```

The first parameter sets the maximum number of characters on a single line. The default value is 80 corresponding roughly to the width of a standard text document.

The second parameter sets the maximum number of terms per line; a term means a sign, a coefficient, and a name (for example "`+ 42 elephants`"). The default value is 0, meaning that there is no maximum.

C.2.4.1 Speeding up file reading

If the input file should be read as fast as possible using the least amount of memory, then it is important to tell MOSEK how many non-zeros, variables and constraints the problem contains. These values can be set using the parameters

```
MSK_IPAR_READ_CON
MSK_IPAR_READ_VAR
MSK_IPAR_READ_ANZ
MSK_IPAR_READ_QNZ
```

C.2.4.2 Unnamed constraints

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

Appendix D

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.

D.1 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 to; 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.

D.2 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 + y + x^2 + 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 in quotes [/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]
```

D.2.1 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 the subsection '`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 apecified 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 the subsections '`b`' (linear bounds on variables) and '`cone`' (quadratic cone).
  - `[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

    $$-10 \leq x, y \leq 10. \tag{D.1}$$

  - `[cone]`. Currently, the supported cones are the *quadratic cone* and the *rotated quadratic cone* A conic constraint is defined as a set of variables which belongs to a single unique cone.

    A quadratic cone of n variables $x_1, \ldots, x_n$ defines a constraint of the form

    $$x_1^2 > \sum_{i=2}^{n} x_i^2.$$

    A rotated quadratic cone of n variables $x_1, \ldots, x_n$ defines a constraint of the form

    $$x_1 x_2 > \sum_{i=3}^{n} x_i^2.$$

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 quad]  x,y,z,w  [/cone] # quadratic cone
  [cone rquad] x,y,z,w  [/cone] # rotated quadratic cone
[/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 values.
- `[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 in a subsection is defined as follows:

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

  where `ITEM` may be replaced by `numvar` (number of variables), `numcon` (number of linear/quadratic constraints), `numanz` (number if linear non-zeros in constraints) and `numqnz` (number of quadratic non-zeros in constraints).
- `[solutions]` This section can contain a number of full or partial solutions to a problem, each inside a `[solution]`-section. The syntax is

  ```
  [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 thing is always to set to status `UNKNOWN`.

A `[solution]`-section contains `[con]` and `[var]` sections. Each `[con]` and `[var]` section defines solution values for a single variable or constraint, each value written as

```
KEYWORD=value
```

where `KEYWORD` defines a solution item and `value` defines its 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 variable associated with its lower bound.
- `su` Defines the level of the 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` and `lvl`, and optionally `sl`, `su` and `sn`.

A `[con]` section should always contain `sk` and `lvl`, and optionally `sl`, `su` and `y`.

- `[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.

D.2.2 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]+)?
```

D.2.3 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_unqouted_name
another_name{123}
'single qouted name'
"double qouted name"
"name with \"qoute\" in it"
"name with []s in it"
```

D.3 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]
```

D.4 Writing OPF files from MOSEK

To write an OPF file set the parameter `MSK_IPAR_WRITE_DATA_FORMAT` to `MSK_DATA_FORMAT_OP` as this ensures that `OPF` format is used. Then modify the following parameters to define what the file should contain:

- `MSK_IPAR_OPF_WRITE_HEADER`, include a small header with comments.
- `MSK_IPAR_OPF_WRITE_HINTS`, include hints about the size of the problem.
- `MSK_IPAR_OPF_WRITE_PROBLEM`, include the problem itself — objective, constraints and bounds.
- `MSK_IPAR_OPF_WRITE_SOLUTIONS`, include solutions if they are defined. If this is off, no solutions are included.
- `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_PARAMETERS`, include all parameter settings.

D.5 Examples

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

D.5.1 Linear example `lo1.opf`

Consider the example:

$$\begin{array}{lrcrcl} \text{minimize} & -10x_1 & & -9x_2, & & \\ \text{subject to} & 7/10x_1 & + & 1x_2 & \leq & 630, \\ & 1/2x_1 & + & 5/6x_2 & \leq & 600, \\ & 1x_1 & + & 2/3x_2 & \leq & 708, \\ & 1/10x_1 & + & 1/4x_2 & \leq & 135, \\ & x_1, & & x_2 & \geq & 0. \end{array} \tag{D.2}$$

In the `OPF` format the example is displayed as shown below:

```
[comment]
  Example lo1.mps converted to OPF.
[/comment]

[hints]
  # Give a hint about the size of the different elements in the problem.
  # These need only be estimates, but in this case they are exact.
  [hint NUMVAR] 2 [/hint]
  [hint NUMCON] 4 [/hint]
  [hint NUMANZ] 8 [/hint]
[/hints]

[variables]
  # All variables that will appear in the problem
  x1 x2
[/variables]

[objective minimize ’obj’]
   - 10 x1 - 9 x2
[/objective]

[constraints]
  [con ’c1’]  0.7 x1 +              x2 <= 630 [/con]
  [con ’c2’]  0.5 x1 + 0.8333333333 x2 <= 600 [/con]
  [con ’c3’]      x1 + 0.66666667   x2 <= 708 [/con]
  [con ’c4’]  0.1 x1 + 0.25         x2 <= 135 [/con]
[/constraints]

[bounds]
  # By default all variables are free. The following line will
  # change this to all variables being nonnegative.
  [b] 0 <= * [/b]
[/bounds]
```

D.5.2 Quadratic example `qo1.opf`

An example of a quadratic optimization problem is

$$\begin{array}{lccc} \text{minimize} & & & x_1^2 + 0.1x_2^2 + x_3^2 - x_1x_3 - x_2 \\ \text{subject to} & 1 & \leq & x_1 + x_2 + x_3, \\ & & & x \geq 0. \end{array} \tag{D.3}$$

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

```
[comment]
  Example qo1.mps conterted to OPF.
[/comment]

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

[variables]
```

```
  x1 x2 x3
[/variables]

[objective minimize ’obj’]
   # The quadratic terms are often multiplied by 1/2,
   # but this is not required.

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

[constraints]
  [con ’c1’] 1 <= x1 + x2 + x3 [/con]
[/constraints]

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

D.5.3 Conic quadratic example `cqo1.opf`

Consider the example:

$$
\begin{array}{lrcl}
\text{minimize} & 1x_1 + 2x_2 & & \\
\text{subject to} & 2x_3 + 4x_4 & = & 5, \\
 & x_5^2 & \leq & 2x_1x_3, \\
 & x_6^2 & \leq & 2x_2x_4, \\
 & x_5 & = & 1, \\
 & x_6 & = & 1, \\
 & x \geq 0. & &
\end{array}
\tag{D.4}
$$

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.

```
[comment]
  Example cqo1.mps conterted to OPF.
[/comment]

[hints]
  [hint NUMVAR] 6 [/hint]
  [hint NUMCON] 1 [/hint]
  [hint NUMANZ] 2 [/hint]
[/hints]

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

[objective minimize ’obj’]
   x1 + 2 x2
[/objective]

[constraints]
  [con ’c1’]  2 x3 + 4 x4 = 5 [/con]
[/constraints]

[bounds]
```

```
  # We let all variables default to the positive orthant
  [b] 0 <= * [/b]
  # ... and change those that differ from the default.
  [b] x5,x6 = 1 [/b]

  # We define two rotated quadratic cones

  # k1: 2 x1 * x3 >= x5^2
  [cone rquad 'k1'] x1, x3, x5 [/cone]

  # k2: 2 x2 * x4 >= x6^2
  [cone rquad 'k2'] x2, x4, x6 [/cone]
[/bounds]
```

D.5.4 Mixed integer example `milo1.opf`

Consider the mixed integer problem:

$$\begin{array}{lrcl} \text{maximize} & x_0 + 0.64x_1 & & \\ \text{subject to} & 50x_0 + 31x_1 & \leq & 250, \\ & 3x_0 - 2x_1 & \geq & -4, \\ & x_0, x_1 \geq 0 & & \text{and integer} \end{array} \tag{D.5}$$

This can be implemented in `OPF` with:

```
[comment]
   Written by MOSEK version 5.0.0.7
   Date 20-11-106
   Time 14:42:24
[/comment]

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

[variables disallow_new_variables]
  x1 x2
[/variables]

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

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

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

[integer]
  x1 x2
```

```
[/integer]
```

Appendix E

The XML (OSiL) format

MOSEK can write data in the standard `OSiL` xml format. For a definition of the `OSiL` format please see `http://www.optimizationservices.org/`. Only linear constraints (possibly with integer variables) are supported. By default output files with the extension `.xml` are written in the `OSiL` format.

The parameter `MSK_IPAR_WRITE_XML_MODE` controls if the linear coefficients in the A matrix are written in row or column order.

Appendix F

The solution file format

MOSEK provides one or two solution files depending on the problem type and the optimizer used. If a problem is optimized using the interior-point optimizer and no basis identification is required, then a file named `probname.sol` is provided. `probname` is the name of the problem and `.sol` is the file extension. If the problem is optimized using the simplex optimizer or basis identification is performed, then a file named `probname.bas` is created presenting the optimal basis solution. Finally, if the problem contains integer constrained variables then a file named `probname.int` is created. It contains the integer solution.

F.1 The basic and interior solution files

In general both the interior-point and the basis 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 statuses are shown. In this case the information shows that the problem is primal and dual feasible and the solution is optimal. Next the primal and dual objective values are displayed.

`CONSTRAINTS` Subsequently in the constraint section the following information is listed for each constraint:

- `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 F.1 the possible values of the status keys and their interpretation are shown.

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.

Table F.1: Status keys.

`ACTIVITY` Given the ith constraint on the form

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

then activity denote the quantity $\sum_{j=1}^{n} a_{ij} x_j^*$, where x^* is the value for the x solution.

`LOWER LIMIT` Is the quantity l_i^c (see (F.1)).

`UPPER LIMIT` Is the quantity u_i^c (see (F.1)).

`DUAL LOWER` Is the dual multiplier corresponding to the lower limit on the constraint.

`DUAL UPPER` Is the dual multiplier corresponding to the upper limit on the constraint.

`VARIABLES` The last section of the solution report lists information for the variables. This information has a similar interpretation as for the constraints. However, the column with the header `[CONIC DUAL]` is only included for problems having one or more conic constraints. This column shows the dual variables corresponding to the conic constraints.

F.2 The integer solution file

The integer solution is equivalent to the basic and interior solution files except that no dual information is included.

Appendix G

The ORD file format

An ORD formatted file specifies in which order the mixed integer optimizer branches on variables. The format of an ORD file is shown in Figure G.1. In the figure names in capitals are keywords of the ORD format, whereas names in brackets are custom names or values. The `??` is an optional key specifying the preferred branching direction. The possible keys are `DN` and `UP` which indicate that down or up is the preferred branching direction respectively. The branching direction key is optional and is left blank the mixed integer optimizer will decide whether to branch up or down.

```
*        1         2         3         4         5         6
*23456789012345678901234567890123456789012345678901234567890
NAME          [name]
 ?? [vname1]              [value1]
ENDATA
```

Figure G.1: The standard ORD format.

G.1 An example

A concrete example of a ORD file is presented below:

```
NAME          EXAMPLE
 DN x1                 2
 UP x2                 1
    x3                 10
ENDATA
```

This implies that the priorities 2, 1, and 10 are assigned to variable `x1`, `x2`, and `x3` respectively. The higher the priority value assigned to a variable the earlier the mixed integer optimizer will branch on that variable. The key `DN` implies that the mixed integer optimizer first will branch down on variable whereas the key `UP` implies that the mixed integer optimizer will first branch up on a variable.

If no branch direction is specified for a variable then the mixed integer optimizer will automatically choose the branching direction for that variable. Similarly, if no priority is assigned to a variable then it is automatically assigned the priority of 0.

Appendix H

Parameters reference

Subsequently all parameters that are in MOSEK parameter database is presented. For each parameter their name, purpose, type, default value etc. are presented.

H.1 Parameter groups

Parameters grouped by meaning and functionality.

H.1.1 Logging parameters.

- `MSK_IPAR_LOG` 200
  Controls the amount of log information.

- `MSK_IPAR_LOG_BI` 201
  Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.

- `MSK_IPAR_LOG_BI_FREQ` 201
  Controls the logging frequency.

- `MSK_IPAR_LOG_CONCURRENT` 201
  Controls amount of output printed by the concurrent optimizer.

- `MSK_IPAR_LOG_CUT_SECOND_OPT` 201
  Controls the reduction in the log levels for the second and any subsequent optimizations.

- `MSK_IPAR_LOG_FACTOR` 202
  If turned on, then the factor log lines are added to the log.

- `MSK_IPAR_LOG_FEASREPAIR` 202
  Controls the amount of output printed when performing feasibility repair.

- `MSK_IPAR_LOG_FILE` 202
  If turned on, then some log info is printed when a file is written or read.

- `MSK_IPAR_LOG_HEAD` 203
  If turned on, then a header line is added to the log.

- `MSK_IPAR_LOG_INFEAS_ANA` 203
  Controls log level for the infeasibility analyzer.

- `MSK_IPAR_LOG_INTPNT` 203
  Controls the amount of log information from the interior-point optimizers.

- `MSK_IPAR_LOG_MIO` 203
  Controls the amount of log information from the mixed-integer optimizers.

- `MSK_IPAR_LOG_MIO_FREQ` 204
  The mixed integer solver logging frequency.

- `MSK_IPAR_LOG_NONCONVEX` 204
  Controls amount of output printed by the nonconvex optimizer.

- `MSK_IPAR_LOG_OPTIMIZER` 204
  Controls the amount of general optimizer information that is logged.

- `MSK_IPAR_LOG_ORDER` 204
  If turned on, then factor lines are added to the log.

- `MSK_IPAR_LOG_PARAM` 205
  Controls the amount of information printed out about parameter changes.

- `MSK_IPAR_LOG_PRESOLVE` 205
  Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.

- `MSK_IPAR_LOG_RESPONSE` 205
  Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.

- `MSK_IPAR_LOG_SENSITIVITY` 205
  Control logging in sensitivity analyzer.

- `MSK_IPAR_LOG_SENSITIVITY_OPT` 206
  Control logging in sensitivity analyzer.

- `MSK_IPAR_LOG_SIM` 206
  Controls the amount of log information from the simplex optimizers.

- `MSK_IPAR_LOG_SIM_FREQ` 206
  Controls simplex logging frequency.

- `MSK_IPAR_LOG_SIM_NETWORK_FREQ` 207
  Controls the network simplex logging frequency.

- `MSK_IPAR_LOG_STORAGE` 207
  Controls the memory related log information.

H.1.2 Basis identification parameters.

- `MSK_IPAR_BI_CLEAN_OPTIMIZER` 187
  Controls which simplex optimizer is used in the clean-up phase.

- `MSK_IPAR_BI_IGNORE_MAX_ITER` 188
  Turns on basis identification in case the interior-point optimizer is terminated due to maximum number of iterations.

- `MSK_IPAR_BI_IGNORE_NUM_ERROR` 188
  Turns on basis identification in case the interior-point optimizer is terminated due to a numerical problem.

- `MSK_DPAR_BI_LU_TOL_REL_PIV` 161
  Relative pivot tolerance used in the LU factorization in the basis identification procedure.

- `MSK_IPAR_BI_MAX_ITERATIONS` 188
  Maximum number of iterations after basis identification.

- `MSK_IPAR_INTPNT_BASIS` 194
  Controls whether basis identification is performed.

- `MSK_IPAR_LOG_BI` 201
  Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.

- `MSK_IPAR_LOG_BI_FREQ` 201
  Controls the logging frequency.

H.1.3 The Interior-point method parameters.

Parameters defining the behavior of the interior-point method for linear, conic and convex problems.

- `MSK_IPAR_BI_IGNORE_MAX_ITER` 188
  Turns on basis identification in case the interior-point optimizer is terminated due to maximum number of iterations.

- `MSK_IPAR_BI_IGNORE_NUM_ERROR` 188
  Turns on basis identification in case the interior-point optimizer is terminated due to a numerical problem.

- `MSK_IPAR_INTPNT_BASIS` 194
  Controls whether basis identification is performed.

- `MSK_DPAR_INTPNT_CO_TOL_DFEAS` 163
  Dual feasibility tolerance used by the conic interior-point optimizer.

- `MSK_DPAR_INTPNT_CO_TOL_INFEAS` 164
  Infeasibility tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_MU_RED` 164
  Optimality tolerance for the conic solver.

- MSK_DPAR_INTPNT_CO_TOL_NEAR_REL 164
  Optimality tolerance for the conic solver.
- MSK_DPAR_INTPNT_CO_TOL_PFEAS 164
  Primal feasibility tolerance used by the conic interior-point optimizer.
- MSK_DPAR_INTPNT_CO_TOL_REL_GAP 165
  Relative gap termination tolerance used by the conic interior-point optimizer.
- MSK_IPAR_INTPNT_DIFF_STEP 195
  Controls whether different step sizes are allowed in the primal and dual space.
- MSK_IPAR_INTPNT_MAX_ITERATIONS 195
  Controls the maximum number of iterations allowed in the interior-point optimizer.
- MSK_IPAR_INTPNT_MAX_NUM_COR 196
  Maximum number of correction steps.
- MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS 196
  Maximum number of steps to be used by the iterative search direction refinement.
- MSK_DPAR_INTPNT_NL_MERIT_BAL 165
  Controls if the complementarity and infeasibility is converging to zero at about equal rates.
- MSK_DPAR_INTPNT_NL_TOL_DFEAS 165
  Dual feasibility tolerance used when a nonlinear model is solved.
- MSK_DPAR_INTPNT_NL_TOL_MU_RED 165
  Relative complementarity gap tolerance.
- MSK_DPAR_INTPNT_NL_TOL_NEAR_REL 166
  Nonlinear solver optimality tolerance parameter.
- MSK_DPAR_INTPNT_NL_TOL_PFEAS 166
  Primal feasibility tolerance used when a nonlinear model is solved.
- MSK_DPAR_INTPNT_NL_TOL_REL_GAP 166
  Relative gap termination tolerance for nonlinear problems.
- MSK_DPAR_INTPNT_NL_TOL_REL_STEP 166
  Relative step size to the boundary for general nonlinear optimization problems.
- MSK_IPAR_INTPNT_OFF_COL_TRH 196
  Controls the aggressiveness of the offending column detection.
- MSK_IPAR_INTPNT_ORDER_METHOD 197
  Controls the ordering strategy.
- MSK_IPAR_INTPNT_REGULARIZATION_USE 197
  Controls whether regularization is allowed.
- MSK_IPAR_INTPNT_SCALING 197
  Controls how the problem is scaled before the interior-point optimizer is used.

- MSK_IPAR_INTPNT_SOLVE_FORM 198
  Controls whether the primal or the dual problem is solved.

- MSK_IPAR_INTPNT_STARTING_POINT 198
  Starting point used by the interior-point optimizer.

- MSK_DPAR_INTPNT_TOL_DFEAS 167
  Dual feasibility tolerance used for linear and quadratic optimization problems.

- MSK_DPAR_INTPNT_TOL_DSAFE 167
  Controls the interior-point dual starting point.

- MSK_DPAR_INTPNT_TOL_INFEAS 167
  Nonlinear solver infeasibility tolerance parameter.

- MSK_DPAR_INTPNT_TOL_MU_RED 167
  Relative complementarity gap tolerance.

- MSK_DPAR_INTPNT_TOL_PATH 168
  interior-point centering aggressiveness.

- MSK_DPAR_INTPNT_TOL_PFEAS 168
  Primal feasibility tolerance used for linear and quadratic optimization problems.

- MSK_DPAR_INTPNT_TOL_PSAFE 168
  Controls the interior-point primal starting point.

- MSK_DPAR_INTPNT_TOL_REL_GAP 168
  Relative gap termination tolerance.

- MSK_DPAR_INTPNT_TOL_REL_STEP 169
  Relative step size to the boundary for linear and quadratic optimization problems.

- MSK_DPAR_INTPNT_TOL_STEP_SIZE 169
  If the step size falls below the value of this parameter, then the interior-point optimizer assumes it is stalled. It it does not not make any progress.

- MSK_IPAR_LOG_CONCURRENT 201
  Controls amount of output printed by the concurrent optimizer.

- MSK_IPAR_LOG_INTPNT 203
  Controls the amount of log information from the interior-point optimizers.

- MSK_IPAR_LOG_PRESOLVE 205
  Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.

H.1.4 Simplex optimizer parameters.

Parameters defining the behavior of the simplex optimizer for linear problems.

- `MSK_DPAR_BASIS_REL_TOL_S` 160
  Maximum relative dual bound violation allowed in an optimal basic solution.
- `MSK_DPAR_BASIS_TOL_S` 160
  Maximum absolute dual bound violation in an optimal basic solution.
- `MSK_DPAR_BASIS_TOL_X` 160
  Maximum absolute primal bound violation allowed in an optimal basic solution.
- `MSK_IPAR_LOG_SIM` 206
  Controls the amount of log information from the simplex optimizers.
- `MSK_IPAR_LOG_SIM_FREQ` 206
  Controls simplex logging frequency.
- `MSK_IPAR_LOG_SIM_MINOR` 206
  Currently not in use.
- `MSK_IPAR_SENSITIVITY_OPTIMIZER` 228
  Controls which optimizer is used for optimal partition sensitivity analysis.
- `MSK_IPAR_SIM_DEGEN` 229
  Controls how aggressive degeneration is approached.
- `MSK_IPAR_SIM_HOTSTART` 230
  Controls the type of hot-start that the simplex optimizer perform.
- `MSK_IPAR_SIM_MAX_ITERATIONS` 231
  Maximum number of iterations that can be used by a simplex optimizer.
- `MSK_IPAR_SIM_MAX_NUM_SETBACKS` 231
  Controls how many setbacks that are allowed within a simplex optimizer.
- `MSK_IPAR_SIM_NETWORK_DETECT_METHOD` 232
  Controls which type of detection method the network extraction should use.
- `MSK_IPAR_SIM_NON_SINGULAR` 232
  Controls if the simplex optimizer ensures a non-singular basis, if possible.
- `MSK_IPAR_SIM_SAVE_LU` 234
  Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.
- `MSK_IPAR_SIM_SCALING` 234
  Controls how the problem is scaled before a simplex optimizer is used.
- `MSK_IPAR_SIM_SOLVE_FORM` 235
  Controls whether the primal or the dual problem is solved by the primal-/dual- simplex optimizer.

- `MSK_IPAR_SIM_STABILITY_PRIORITY` 235
  Controls how high priority the numerical stability should be given.

- `MSK_IPAR_SIM_SWITCH_OPTIMIZER` 235
  Controls the simplex behavior.

- `MSK_DPAR_SIMPLEX_ABS_TOL_PIV` 175
  Absolute pivot tolerance employed by the simplex optimizers.

H.1.5 Primal simplex optimizer parameters.

Parameters defining the behavior of the primal simplex optimizer for linear problems.

- `MSK_IPAR_SIM_PRIMAL_CRASH` 232
  Controls the simplex crash.

- `MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION` 233
  Controls how aggressively restricted selection is used.

- `MSK_IPAR_SIM_PRIMAL_SELECTION` 233
  Controls the primal simplex strategy.

H.1.6 Dual simplex optimizer parameters.

Parameters defining the behavior of the dual simplex optimizer for linear problems.

- `MSK_IPAR_SIM_DUAL_CRASH` 229
  Controls whether crashing is performed in the dual simplex optimizer.

- `MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION` 229
  Controls how aggressively restricted selection is used.

- `MSK_IPAR_SIM_DUAL_SELECTION` 230
  Controls the dual simplex strategy.

H.1.7 Network simplex optimizer parameters.

Parameters defining the behavior of the network simplex optimizer for linear problems.

- `MSK_IPAR_LOG_SIM_NETWORK_FREQ` 207
  Controls the network simplex logging frequency.

- `MSK_IPAR_SIM_NETWORK_DETECT` 231
  Level of aggressiveness of network detection.

- `MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART` 232
  Level of aggressiveness of network detection in a simplex hot-start.

- `MSK_IPAR_SIM_REFACTOR_FREQ` 234
  Controls the basis refactoring frequency.

H.1.8 Nonlinear convex method parameters.

Parameters defining the behavior of the interior-point method for nonlinear convex problems.

- `MSK_IPAR_CHECK_CONVEXITY` 189
  Specify the level of convexity check on quadratic problems
- `MSK_DPAR_INTPNT_NL_MERIT_BAL` 165
  Controls if the complementarity and infeasibility is converging to zero at about equal rates.
- `MSK_DPAR_INTPNT_NL_TOL_DFEAS` 165
  Dual feasibility tolerance used when a nonlinear model is solved.
- `MSK_DPAR_INTPNT_NL_TOL_MU_RED` 165
  Relative complementarity gap tolerance.
- `MSK_DPAR_INTPNT_NL_TOL_NEAR_REL` 166
  Nonlinear solver optimality tolerance parameter.
- `MSK_DPAR_INTPNT_NL_TOL_PFEAS` 166
  Primal feasibility tolerance used when a nonlinear model is solved.
- `MSK_DPAR_INTPNT_NL_TOL_REL_GAP` 166
  Relative gap termination tolerance for nonlinear problems.
- `MSK_DPAR_INTPNT_NL_TOL_REL_STEP` 166
  Relative step size to the boundary for general nonlinear optimization problems.
- `MSK_DPAR_INTPNT_TOL_INFEAS` 167
  Nonlinear solver infeasibility tolerance parameter.

H.1.9 The conic interior-point method parameters.

Parameters defining the behavior of the interior-point method for conic problems.

- `MSK_DPAR_INTPNT_CO_TOL_DFEAS` 163
  Dual feasibility tolerance used by the conic interior-point optimizer.
- `MSK_DPAR_INTPNT_CO_TOL_INFEAS` 164
  Infeasibility tolerance for the conic solver.
- `MSK_DPAR_INTPNT_CO_TOL_MU_RED` 164
  Optimality tolerance for the conic solver.
- `MSK_DPAR_INTPNT_CO_TOL_NEAR_REL` 164
  Optimality tolerance for the conic solver.
- `MSK_DPAR_INTPNT_CO_TOL_PFEAS` 164
  Primal feasibility tolerance used by the conic interior-point optimizer.
- `MSK_DPAR_INTPNT_CO_TOL_REL_GAP` 165
  Relative gap termination tolerance used by the conic interior-point optimizer.

H.1.10 The mixed integer optimization parameters.

- `MSK_IPAR_LOG_MIO` 203
  Controls the amount of log information from the mixed-integer optimizers.

- `MSK_IPAR_LOG_MIO_FREQ` 204
  The mixed integer solver logging frequency.

- `MSK_IPAR_MIO_BRANCH_DIR` 208
  Controls whether the mixed integer optimizer is branching up or down by default.

- `MSK_IPAR_MIO_BRANCH_PRIORITIES_USE` 208
  Controls whether branching priorities are used by the mixed integer optimizer.

- `MSK_IPAR_MIO_CONSTRUCT_SOL` 209
  Controls if initial MIP solution should be constructed from value of integer variables.

- `MSK_IPAR_MIO_CONT_SOL` 209
  Controls the meaning of interior-point and basic solutions in MIP problems.

- `MSK_IPAR_MIO_CUT_LEVEL_ROOT` 209
  Controls the cut level employed by the mixed integer optimizer at the root node.

- `MSK_IPAR_MIO_CUT_LEVEL_TREE` 210
  Controls the cut level employed by the mixed integer optimizer in the tree.

- `MSK_DPAR_MIO_DISABLE_TERM_TIME` 170
  Certain termination criterias is disabled within the mixed integer optimizer for period time specified by the parameter.

- `MSK_IPAR_MIO_FEASPUMP_LEVEL` 210
  Controls the feasibility pump heuristic which is used to construct a good initial feasible solution.

- `MSK_IPAR_MIO_HEURISTIC_LEVEL` 211
  Controls the heuristic employed by the mixed integer optimizer to locate an initial integer feasible solution.

- `MSK_DPAR_MIO_HEURISTIC_TIME` 170
  Time limit for the mixed integer heuristics.

- `MSK_IPAR_MIO_KEEP_BASIS` 211
  Controls whether the integer presolve keeps bases in memory.

- `MSK_IPAR_MIO_MAX_NUM_BRANCHES` 211
  Maximum number of branches allowed during the branch and bound search.

- `MSK_IPAR_MIO_MAX_NUM_RELAXS` 212
  Maximum number of relaxations in branch and bound search.

- `MSK_IPAR_MIO_MAX_NUM_SOLUTIONS` 212
  Controls how many feasible solutions the mixed-integer optimizer investigates.

- MSK_DPAR_MIO_MAX_TIME 171
  Time limit for the mixed integer optimizer.

- MSK_DPAR_MIO_MAX_TIME_APRX_OPT 171
  Time limit for the mixed integer optimizer.

- MSK_DPAR_MIO_NEAR_TOL_ABS_GAP 171
  Relaxed absolute optimality tolerance employed by the mixed integer optimizer.

- MSK_DPAR_MIO_NEAR_TOL_REL_GAP 171
  The mixed integer optimizer is terminated when this tolerance is satisfied.

- MSK_IPAR_MIO_NODE_OPTIMIZER 213
  Controls which optimizer is employed at the non-root nodes in the mixed integer optimizer.

- MSK_IPAR_MIO_NODE_SELECTION 213
  Controls the node selection strategy employed by the mixed integer optimizer.

- MSK_IPAR_MIO_PRESOLVE_AGGREGATE 214
  Controls whether problem aggregation is performed in the mixed integer presolve.

- MSK_IPAR_MIO_PRESOLVE_PROBING 214
  Controls whether probing is employed by the mixed integer presolve.

- MSK_IPAR_MIO_PRESOLVE_USE 214
  Controls whether presolve is performed by the mixed integer optimizer.

- MSK_DPAR_MIO_REL_ADD_CUT_LIMITED 172
  Controls cut generation for MIP solver.

- MSK_IPAR_MIO_ROOT_OPTIMIZER 215
  Controls which optimizer is employed at the root node in the mixed integer optimizer.

- MSK_IPAR_MIO_STRONG_BRANCH 215
  The depth from the root in which strong branching is employed.

- MSK_DPAR_MIO_TOL_ABS_GAP 172
  Absolute optimality tolerance employed by the mixed integer optimizer.

- MSK_DPAR_MIO_TOL_ABS_RELAX_INT 172
  Integer constraint tolerance.

- MSK_DPAR_MIO_TOL_REL_GAP 173
  Relative optimality tolerance employed by the mixed integer optimizer.

- MSK_DPAR_MIO_TOL_REL_RELAX_INT 173
  Integer constraint tolerance.

- MSK_DPAR_MIO_TOL_X 173
  Absolute solution tolerance used in mixed-integer optimizer.

H.1.11 Presolve parameters.

- `MSK_IPAR_PRESOLVE_ELIM_FILL` 220
  Maximum amount of fill-in in the elimination phase.

- `MSK_IPAR_PRESOLVE_ELIMINATOR_USE` 220
  Controls whether free or implied free variables are eliminated from the problem.

- `MSK_IPAR_PRESOLVE_LEVEL` 220
  Currently not used.

- `MSK_IPAR_PRESOLVE_LINDEP_USE` 220
  Controls whether the linear constraints are checked for linear dependencies.

- `MSK_IPAR_PRESOLVE_LINDEP_WORK_LIM` 221
  Controls linear dependency check in presolve.

- `MSK_DPAR_PRESOLVE_TOL_AIJ` 174
  Absolute zero tolerance employed for constraint coefficients in the presolve.

- `MSK_DPAR_PRESOLVE_TOL_LIN_DEP` 174
  Controls when a constraint is determined to be linearly dependent.

- `MSK_DPAR_PRESOLVE_TOL_S` 175
  Absolute zero tolerance employed for slack variables in the presolve.

- `MSK_DPAR_PRESOLVE_TOL_X` 175
  Absolute zero tolerance employed for variables in the presolve.

- `MSK_IPAR_PRESOLVE_USE` 221
  Controls whether the presolve is applied to a problem before it is optimized.

H.1.12 Termination criterion parameters.

Parameters which define termination and optimality criteria and related information.

- `MSK_DPAR_BASIS_REL_TOL_S` 160
  Maximum relative dual bound violation allowed in an optimal basic solution.

- `MSK_DPAR_BASIS_TOL_S` 160
  Maximum absolute dual bound violation in an optimal basic solution.

- `MSK_DPAR_BASIS_TOL_X` 160
  Maximum absolute primal bound violation allowed in an optimal basic solution.

- `MSK_IPAR_BI_MAX_ITERATIONS` 188
  Maximum number of iterations after basis identification.

- `MSK_DPAR_INTPNT_CO_TOL_DFEAS` 163
  Dual feasibility tolerance used by the conic interior-point optimizer.

- `MSK_DPAR_INTPNT_CO_TOL_INFEAS` 164
  Infeasibility tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_MU_RED` 164
  Optimality tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_NEAR_REL` 164
  Optimality tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_PFEAS` 164
  Primal feasibility tolerance used by the conic interior-point optimizer.

- `MSK_DPAR_INTPNT_CO_TOL_REL_GAP` 165
  Relative gap termination tolerance used by the conic interior-point optimizer.

- `MSK_IPAR_INTPNT_MAX_ITERATIONS` 195
  Controls the maximum number of iterations allowed in the interior-point optimizer.

- `MSK_DPAR_INTPNT_NL_TOL_DFEAS` 165
  Dual feasibility tolerance used when a nonlinear model is solved.

- `MSK_DPAR_INTPNT_NL_TOL_MU_RED` 165
  Relative complementarity gap tolerance.

- `MSK_DPAR_INTPNT_NL_TOL_NEAR_REL` 166
  Nonlinear solver optimality tolerance parameter.

- `MSK_DPAR_INTPNT_NL_TOL_PFEAS` 166
  Primal feasibility tolerance used when a nonlinear model is solved.

- `MSK_DPAR_INTPNT_NL_TOL_REL_GAP` 166
  Relative gap termination tolerance for nonlinear problems.

- `MSK_DPAR_INTPNT_TOL_DFEAS` 167
  Dual feasibility tolerance used for linear and quadratic optimization problems.

- `MSK_DPAR_INTPNT_TOL_INFEAS` 167
  Nonlinear solver infeasibility tolerance parameter.

- `MSK_DPAR_INTPNT_TOL_MU_RED` 167
  Relative complementarity gap tolerance.

- `MSK_DPAR_INTPNT_TOL_PFEAS` 168
  Primal feasibility tolerance used for linear and quadratic optimization problems.

- `MSK_DPAR_INTPNT_TOL_REL_GAP` 168
  Relative gap termination tolerance.

- `MSK_DPAR_LOWER_OBJ_CUT` 169
  Objective bound.

- `MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH` 169
  Objective bound.

- `MSK_DPAR_MIO_DISABLE_TERM_TIME` 170
  Certain termination criterias is disabled within the mixed integer optimizer for period time specified by the parameter.

- `MSK_IPAR_MIO_MAX_NUM_BRANCHES` 211
  Maximum number of branches allowed during the branch and bound search.

- `MSK_IPAR_MIO_MAX_NUM_SOLUTIONS` 212
  Controls how many feasible solutions the mixed-integer optimizer investigates.

- `MSK_DPAR_MIO_MAX_TIME` 171
  Time limit for the mixed integer optimizer.

- `MSK_DPAR_MIO_NEAR_TOL_REL_GAP` 171
  The mixed integer optimizer is terminated when this tolerance is satisfied.

- `MSK_DPAR_MIO_TOL_REL_GAP` 173
  Relative optimality tolerance employed by the mixed integer optimizer.

- `MSK_DPAR_OPTIMIZER_MAX_TIME` 174
  Solver time limit.

- `MSK_IPAR_SIM_MAX_ITERATIONS` 231
  Maximum number of iterations that can be used by a simplex optimizer.

- `MSK_DPAR_UPPER_OBJ_CUT` 175
  Objective bound.

- `MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH` 176
  Objective bound.

H.1.13 Progress call-back parameters.

- `MSK_DPAR_CALLBACK_FREQ` 161
  Controls progress call-back frequency.

- `MSK_IPAR_SOLUTION_CALLBACK` 237
  Indicates whether solution call-backs will be performed during the optimization.

H.1.14 Non-convex solver parameters.

- `MSK_IPAR_LOG_NONCONVEX` 204
  Controls amount of output printed by the nonconvex optimizer.

- `MSK_IPAR_NONCONVEX_MAX_ITERATIONS` 215
  Maximum number of iterations that can be used by the nonconvex optimizer.

- `MSK_DPAR_NONCONVEX_TOL_FEAS` 173
  Feasibility tolerance used by the nonconvex optimizer.

- `MSK_DPAR_NONCONVEX_TOL_OPT` 174
  Optimality tolerance used by the nonconvex optimizer.

H.1.15 Feasibility repair parameters.

- MSK_DPAR_FEASREPAIR_TOL 163
  Tolerance for constraint enforcing upper bound on sum of weighted violations in feasibility repair.

H.1.16 Optimization system parameters.

Parameters defining the overall solver system environment. This includes system and platform related information and behavior.

- MSK_IPAR_CACHE_SIZE_L1 189
  Specifies the size of the level 1 cache of the processor.

- MSK_IPAR_CACHE_SIZE_L2 189
  Specifies the size of the level 2 cache of the processor.

- MSK_IPAR_CHECK_CTRL_C 190
  Turns ctrl-c check on or off.

- MSK_IPAR_CPU_TYPE 191
  Specifies the CPU type.

- MSK_IPAR_INTPNT_NUM_THREADS 196
  Controls the number of threads employed by the interior-point optimizer.

- MSK_IPAR_LICENSE_CACHE_TIME 199
  Controls the license manager client behavior.

- MSK_IPAR_LICENSE_CHECK_TIME 199
  Controls the license manager client behavior.

- MSK_IPAR_LICENSE_WAIT 200
  Controls if MOSEK should queue for a license if none is available.

- MSK_IPAR_LOG_STORAGE 207
  Controls the memory related log information.

H.1.17 Output information parameters.

- MSK_IPAR_FLUSH_STREAM_FREQ 193
  Controls the stream flushing frequency.

- MSK_IPAR_INFEAS_REPORT_LEVEL 194
  Controls the contents of the infeasibility report.

- MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS 200
  Controls license manager client behavior.

- MSK_IPAR_LOG 200
  Controls the amount of log information.

- `MSK_IPAR_LOG_BI` 201
  Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.

- `MSK_IPAR_LOG_BI_FREQ` 201
  Controls the logging frequency.

- `MSK_IPAR_LOG_CUT_SECOND_OPT` 201
  Controls the reduction in the log levels for the second and any subsequent optimizations.

- `MSK_IPAR_LOG_FACTOR` 202
  If turned on, then the factor log lines are added to the log.

- `MSK_IPAR_LOG_FEASREPAIR` 202
  Controls the amount of output printed when performing feasibility repair.

- `MSK_IPAR_LOG_FILE` 202
  If turned on, then some log info is printed when a file is written or read.

- `MSK_IPAR_LOG_HEAD` 203
  If turned on, then a header line is added to the log.

- `MSK_IPAR_LOG_INFEAS_ANA` 203
  Controls log level for the infeasibility analyzer.

- `MSK_IPAR_LOG_INTPNT` 203
  Controls the amount of log information from the interior-point optimizers.

- `MSK_IPAR_LOG_MIO` 203
  Controls the amount of log information from the mixed-integer optimizers.

- `MSK_IPAR_LOG_MIO_FREQ` 204
  The mixed integer solver logging frequency.

- `MSK_IPAR_LOG_NONCONVEX` 204
  Controls amount of output printed by the nonconvex optimizer.

- `MSK_IPAR_LOG_OPTIMIZER` 204
  Controls the amount of general optimizer information that is logged.

- `MSK_IPAR_LOG_ORDER` 204
  If turned on, then factor lines are added to the log.

- `MSK_IPAR_LOG_PARAM` 205
  Controls the amount of information printed out about parameter changes.

- `MSK_IPAR_LOG_RESPONSE` 205
  Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.

- `MSK_IPAR_LOG_SENSITIVITY` 205
  Control logging in sensitivity analyzer.

- `MSK_IPAR_LOG_SENSITIVITY_OPT` 206
  Control logging in sensitivity analyzer.

- `MSK_IPAR_LOG_SIM` 206
  Controls the amount of log information from the simplex optimizers.

- `MSK_IPAR_LOG_SIM_FREQ` 206
  Controls simplex logging frequency.

- `MSK_IPAR_LOG_SIM_MINOR` 206
  Currently not in use.

- `MSK_IPAR_LOG_SIM_NETWORK_FREQ` 207
  Controls the network simplex logging frequency.

- `MSK_IPAR_LOG_STORAGE` 207
  Controls the memory related log information.

- `MSK_IPAR_MAX_NUM_WARNINGS` 208
  Waning level. A higher value results in more warnings.

- `MSK_IPAR_WARNING_LEVEL` 237
  Warning level.

H.1.18 Extra information about the optimization problem.

- `MSK_IPAR_OBJECTIVE_SENSE` 216
  If the objective sense for the task is undefined, then the value of this parameter is used as the default objective sense.

H.1.19 Overall solver parameters.

- `MSK_IPAR_BI_CLEAN_OPTIMIZER` 187
  Controls which simplex optimizer is used in the clean-up phase.

- `MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS` 190
  The maximum number of simultaneous optimizations that will be started by the concurrent optimizer.

- `MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX` 191
  Priority of the dual simplex algorithm when selecting solvers for concurrent optimization.

- `MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX` 191
  Priority of the free simplex optimizer when selecting solvers for concurrent optimization.

- `MSK_IPAR_CONCURRENT_PRIORITY_INTPNT` 191
  Priority of the interior-point algorithm when selecting solvers for concurrent optimization.

- `MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX` 191
  Priority of the primal simplex algorithm when selecting solvers for concurrent optimization.

- MSK_IPAR_DATA_CHECK 192
  Enable data checking for debug purposes.
- MSK_IPAR_FEASREPAIR_OPTIMIZE 192
  Controls which type of feasibility analysis is to be performed.
- MSK_IPAR_INFEAS_PREFER_PRIMAL 193
  Controls which certificate is used if both primal- and dual- certificate of infeasibility is available.
- MSK_IPAR_LICENSE_WAIT 200
  Controls if MOSEK should queue for a license if none is available.
- MSK_IPAR_MIO_CONT_SOL 209
  Controls the meaning of interior-point and basic solutions in MIP problems.
- MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER 211
  Controls the size of the local search space when doing local branching.
- MSK_IPAR_MIO_MODE 212
  Turns on/off the mixed integer mode.
- MSK_IPAR_OPTIMIZER 218
  Controls which optimizer is used to optimize the task.
- MSK_IPAR_PRESOLVE_LEVEL 220
  Currently not used.
- MSK_IPAR_PRESOLVE_USE 221
  Controls whether the presolve is applied to a problem before it is optimized.
- MSK_IPAR_SENSITIVITY_ALL 228
  Controls sensitivity report behavior.
- MSK_IPAR_SENSITIVITY_OPTIMIZER 228
  Controls which optimizer is used for optimal partition sensitivity analysis.
- MSK_IPAR_SENSITIVITY_TYPE 228
  Controls which type of sensitivity analysis is to be performed.
- MSK_IPAR_SOLUTION_CALLBACK 237
  Indicates whether solution call-backs will be performed during the optimization.

H.1.20 Behavior of the optimization task.

Parameters defining the behavior of an optimization task when loading data.

- MSK_IPAR_ALLOC_ADD_QNZ 187
  Controls how the quadratic matrixes are extended.
- MSK_SPAR_FEASREPAIR_NAME_PREFIX 247
  Feasibility repair name prefix.

- MSK_SPAR_FEASREPAIR_NAME_SEPARATOR 247
  Feasibility repair name separator.
- MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL 247
  Feasibility repair name violation name.
- MSK_IPAR_MAXNUMANZ_DOUBLE_TRH 208
  Controls how the constraint matrix is extended.
- MSK_IPAR_READ_ADD_ANZ 221
  Controls how the constraint matrix is extended.
- MSK_IPAR_READ_ADD_CON 221
  Additional number of constraints that is made room for in the problem.
- MSK_IPAR_READ_ADD_CONE 222
  Additional number of conic constraints that is made room for in the problem.
- MSK_IPAR_READ_ADD_QNZ 222
  Controls how the quadratic matrixes are extended.
- MSK_IPAR_READ_ADD_VAR 222
  Additional number of variables that is made room for in the problem.
- MSK_IPAR_READ_ANZ 222
  Controls the expected number of constraint non-zeros.
- MSK_IPAR_READ_CON 223
  Controls the expected number of constraints.
- MSK_IPAR_READ_CONE 223
  Controls the expected number of conic constraints.
- MSK_IPAR_READ_QNZ 227
  Controls the expected number of quadratic non-zeros.
- MSK_IPAR_READ_TASK_IGNORE_PARAM 227
  Controls what information is used from the task files.
- MSK_IPAR_READ_VAR 227
  Controls the expected number of variables.
- MSK_IPAR_WRITE_TASK_INC_SOL 244
  Controls whether the solutions are stored in the task file too.

H.1.21 Data input/output parameters.

Parameters defining the behavior of data readers and writers.

- MSK_SPAR_BAS_SOL_FILE_NAME 246
  Name of the bas solution file.

- `MSK_SPAR_DATA_FILE_NAME` ... 246
  Data are read and written to this file.
- `MSK_SPAR_DEBUG_FILE_NAME` ... 247
  MOSEK debug file.
- `MSK_IPAR_INFEAS_REPORT_AUTO` ... 193
  Turns the feasibility report on or off.
- `MSK_SPAR_INT_SOL_FILE_NAME` ... 248
  Name of the int solution file.
- `MSK_SPAR_ITR_SOL_FILE_NAME` ... 248
  Name of the itr solution file.
- `MSK_IPAR_LOG_FILE` ... 202
  If turned on, then some log info is printed when a file is written or read.
- `MSK_IPAR_LP_WRITE_IGNORE_INCOMPATIBLE_ITEMS` ... 207
  Controls the result of writing a problem containing incompatible items to an LP file.
- `MSK_IPAR_OPF_MAX_TERMS_PER_LINE` ... 216
  The maximum number of terms (linear and quadratic) per line when an OPF file is written.
- `MSK_IPAR_OPF_WRITE_HEADER` ... 216
  Write a text header with date and MOSEK version in an OPF file.
- `MSK_IPAR_OPF_WRITE_HINTS` ... 216
  Write a hint section with problem dimensions in the beginning of an OPF file.
- `MSK_IPAR_OPF_WRITE_PARAMETERS` ... 217
  Write a parameter section in an OPF file.
- `MSK_IPAR_OPF_WRITE_PROBLEM` ... 217
  Write objective, constraints, bounds etc. to an OPF file.
- `MSK_IPAR_OPF_WRITE_SOL_BAS` ... 217
  Controls what is written to the OPF files.
- `MSK_IPAR_OPF_WRITE_SOL_ITG` ... 218
  Controls what is written to the OPF files.
- `MSK_IPAR_OPF_WRITE_SOL_ITR` ... 218
  Controls what is written to the OPF files.
- `MSK_IPAR_OPF_WRITE_SOLUTIONS` ... 218
  Enable inclusion of solutions in the OPF files.
- `MSK_SPAR_PARAM_COMMENT_SIGN` ... 248
  Solution file comment character.
- `MSK_IPAR_PARAM_READ_CASE_NAME` ... 219
  If turned on, then names in the parameter file are case sensitive.

- `MSK_SPAR_PARAM_READ_FILE_NAME` 248
  Modifications to the parameter database is read from this file.
- `MSK_IPAR_PARAM_READ_IGN_ERROR` 219
  If turned on, then errors in paramter settings is ignored.
- `MSK_SPAR_PARAM_WRITE_FILE_NAME` 249
  The parameter database is written to this file.
- `MSK_IPAR_READ_ADD_ANZ` 221
  Controls how the constraint matrix is extended.
- `MSK_IPAR_READ_ADD_CON` 221
  Additional number of constraints that is made room for in the problem.
- `MSK_IPAR_READ_ADD_CONE` 222
  Additional number of conic constraints that is made room for in the problem.
- `MSK_IPAR_READ_ADD_QNZ` 222
  Controls how the quadratic matrixes are extended.
- `MSK_IPAR_READ_ADD_VAR` 222
  Additional number of variables that is made room for in the problem.
- `MSK_IPAR_READ_ANZ` 222
  Controls the expected number of constraint non-zeros.
- `MSK_IPAR_READ_CON` 223
  Controls the expected number of constraints.
- `MSK_IPAR_READ_CONE` 223
  Controls the expected number of conic constraints.
- `MSK_IPAR_READ_DATA_COMPRESSED` 223
  Controls the input file decompression.
- `MSK_IPAR_READ_DATA_FORMAT` 223
  Format of the data file to be read.
- `MSK_IPAR_READ_KEEP_FREE_CON` 224
  Controls whether the free constraints are included in the problem.
- `MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU` 224
  Controls how the LP files are interpreted.
- `MSK_IPAR_READ_LP_QUOTED_NAMES` 224
  If a name is in quotes when reading an LP file, the quotes will be removed.
- `MSK_SPAR_READ_MPS_BOU_NAME` 249
  Name of the BOUNDS vector used. An empty name means that the first BOUNDS vector is used.

- `MSK_IPAR_READ_MPS_FORMAT` 225
  Controls how strictly the MPS file reader interprets the MPS format.

- `MSK_IPAR_READ_MPS_KEEP_INT` 225
  Controls if integer constraints are read.

- `MSK_SPAR_READ_MPS_OBJ_NAME` 249
  Objective name in the MPS file.

- `MSK_IPAR_READ_MPS_OBJ_SENSE` 225
  Controls the MPS format extensions.

- `MSK_IPAR_READ_MPS_QUOTED_NAMES` 226
  Controls the MPS format extensions.

- `MSK_SPAR_READ_MPS_RAN_NAME` 249
  Name of the RANGE vector used. An empty name means that the first RANGE vector is used.

- `MSK_IPAR_READ_MPS_RELAX` 226
  Controls the meaning of integer constraints.

- `MSK_SPAR_READ_MPS_RHS_NAME` 250
  Name of the RHS used. An empty name means that the first RHS vector is used.

- `MSK_IPAR_READ_MPS_WIDTH` 226
  Controls the maximal number of chars allowed in one line of the MPS file.

- `MSK_IPAR_READ_Q_MODE` 226
  Controls how the Q matrices are read from the MPS file.

- `MSK_IPAR_READ_QNZ` 227
  Controls the expected number of quadratic non-zeros.

- `MSK_IPAR_READ_TASK_IGNORE_PARAM` 227
  Controls what information is used from the task files.

- `MSK_IPAR_READ_VAR` 227
  Controls the expected number of variables.

- `MSK_SPAR_SENSITIVITY_FILE_NAME` 250
  Sensitivity report file name.

- `MSK_SPAR_SENSITIVITY_RES_FILE_NAME` 250
  Name of the sensitivity report output file.

- `MSK_SPAR_SOL_FILTER_XC_LOW` 251
  Solution file filter.

- `MSK_SPAR_SOL_FILTER_XC_UPR` 251
  Solution file filter.

- `MSK_SPAR_SOL_FILTER_XX_LOW` 251
  Solution file filter.

- MSK_SPAR_SOL_FILTER_XX_UPR 251
  Solution file filter.

- MSK_IPAR_SOL_QUOTED_NAMES 236
  Controls the solution file format.

- MSK_IPAR_SOL_READ_NAME_WIDTH 236
  Controls the input solution file format.

- MSK_IPAR_SOL_READ_WIDTH 237
  Controls the input solution file format.

- MSK_SPAR_STAT_FILE_NAME 252
  Statistics file name.

- MSK_SPAR_STAT_KEY 252
  Key used when writing the summary file.

- MSK_SPAR_STAT_NAME 252
  Name used when writing the statistics file.

- MSK_IPAR_WRITE_BAS_CONSTRAINTS 237
  Controls the basic solution file format.

- MSK_IPAR_WRITE_BAS_HEAD 238
  Controls the basic solution file format.

- MSK_IPAR_WRITE_BAS_VARIABLES 238
  Controls the basic solution file format.

- MSK_IPAR_WRITE_DATA_COMPRESSED 238
  Controls output file compression.

- MSK_IPAR_WRITE_DATA_FORMAT 239
  Controls the output file problem format.

- MSK_IPAR_WRITE_DATA_PARAM 239
  Controls output file data.

- MSK_IPAR_WRITE_FREE_CON 239
  Controls the output file data.

- MSK_IPAR_WRITE_GENERIC_NAMES 240
  Controls the output file data.

- MSK_IPAR_WRITE_GENERIC_NAMES_IO 240
  Index origin used in generic names.

- MSK_IPAR_WRITE_INT_CONSTRAINTS 240
  Controls the integer solution file format.

- MSK_IPAR_WRITE_INT_HEAD 240
  Controls the integer solution file format.

- `MSK_IPAR_WRITE_INT_VARIABLES` 241
  Controls the integer solution file format.

- `MSK_SPAR_WRITE_LP_GEN_VAR_NAME` 252
  Added variable names in the LP files.

- `MSK_IPAR_WRITE_LP_LINE_WIDTH` 241
  Controls the LP output file format.

- `MSK_IPAR_WRITE_LP_QUOTED_NAMES` 241
  Controls LP output file format.

- `MSK_IPAR_WRITE_LP_STRICT_FORMAT` 241
  Controls whether LP output files satisfy the LP format strictly.

- `MSK_IPAR_WRITE_LP_TERMS_PER_LINE` 242
  Controls the LP output file format.

- `MSK_IPAR_WRITE_MPS_INT` 242
  Controls the output file data.

- `MSK_IPAR_WRITE_MPS_OBJ_SENSE` 242
  Controls the output file data.

- `MSK_IPAR_WRITE_MPS_QUOTED_NAMES` 242
  Controls the output file data.

- `MSK_IPAR_WRITE_MPS_STRICT` 243
  Controls the output MPS file format.

- `MSK_IPAR_WRITE_PRECISION` 243
  Controls data precision employed in when writing an MPS file.

- `MSK_IPAR_WRITE_SOL_CONSTRAINTS` 243
  Controls the solution file format.

- `MSK_IPAR_WRITE_SOL_HEAD` 244
  Controls solution file format.

- `MSK_IPAR_WRITE_SOL_VARIABLES` 244
  Controls the solution file format.

- `MSK_IPAR_WRITE_TASK_INC_SOL` 244
  Controls whether the solutions are stored in the task file too.

- `MSK_IPAR_WRITE_XML_MODE` 244
  Controls if linear coefficients should be written by row or column when writing in the XML file format.

H.1.22 Solution input/output parameters.

Parameters defining the behavior of solution reader and writer.

- MSK_SPAR_BAS_SOL_FILE_NAME 246
  Name of the bas solution file.
- MSK_IPAR_INFEAS_REPORT_AUTO 193
  Turns the feasibility report on or off.
- MSK_SPAR_INT_SOL_FILE_NAME 248
  Name of the int solution file.
- MSK_SPAR_ITR_SOL_FILE_NAME 248
  Name of the itr solution file.
- MSK_IPAR_SOL_FILTER_KEEP_BASIC 235
  Controls the license manager client behavior.
- MSK_IPAR_SOL_FILTER_KEEP_RANGED 236
  Control the contents of the solution files.
- MSK_SPAR_SOL_FILTER_XC_LOW 251
  Solution file filter.
- MSK_SPAR_SOL_FILTER_XC_UPR 251
  Solution file filter.
- MSK_SPAR_SOL_FILTER_XX_LOW 251
  Solution file filter.
- MSK_SPAR_SOL_FILTER_XX_UPR 251
  Solution file filter.
- MSK_IPAR_SOL_QUOTED_NAMES 236
  Controls the solution file format.
- MSK_IPAR_SOL_READ_NAME_WIDTH 236
  Controls the input solution file format.
- MSK_IPAR_SOL_READ_WIDTH 237
  Controls the input solution file format.
- MSK_IPAR_WRITE_BAS_CONSTRAINTS 237
  Controls the basic solution file format.
- MSK_IPAR_WRITE_BAS_HEAD 238
  Controls the basic solution file format.
- MSK_IPAR_WRITE_BAS_VARIABLES 238
  Controls the basic solution file format.

- `MSK_IPAR_WRITE_INT_CONSTRAINTS` 240
  Controls the integer solution file format.

- `MSK_IPAR_WRITE_INT_HEAD` 240
  Controls the integer solution file format.

- `MSK_IPAR_WRITE_INT_VARIABLES` 241
  Controls the integer solution file format.

- `MSK_IPAR_WRITE_SOL_CONSTRAINTS` 243
  Controls the solution file format.

- `MSK_IPAR_WRITE_SOL_HEAD` 244
  Controls solution file format.

- `MSK_IPAR_WRITE_SOL_VARIABLES` 244
  Controls the solution file format.

H.1.23 Infeasibility report parameters.

- `MSK_IPAR_INFEAS_GENERIC_NAMES` 193
  Controls the contents of the infeasibility report.

- `MSK_IPAR_INFEAS_REPORT_LEVEL` 194
  Controls the contents of the infeasibility report.

- `MSK_IPAR_LOG_INFEAS_ANA` 203
  Controls log level for the infeasibility analyzer.

H.1.24 License manager parameters.

- `MSK_IPAR_LICENSE_ALLOW_OVERUSE` 198
  Controls if license overuse is allowed when caching licenses

- `MSK_IPAR_LICENSE_CACHE_TIME` 199
  Controls the license manager client behavior.

- `MSK_IPAR_LICENSE_CHECK_TIME` 199
  Controls the license manager client behavior.

- `MSK_IPAR_LICENSE_DEBUG` 199
  Controls the license manager client debugging behavior.

- `MSK_IPAR_LICENSE_PAUSE_TIME` 199
  Controls license manager client behavior.

- `MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS` 200
  Controls license manager client behavior.

- `MSK_IPAR_LICENSE_WAIT` 200
  Controls if MOSEK should queue for a license if none is available.

H.1.25 Data check parameters.

These parameters defines data checking settings and problem data tolerances, i.e. which values are rounded to 0 or infinity, and which values are large or small enough to produce a warning.

- MSK_IPAR_CHECK_CONVEXITY 189
  Specify the level of convexity check on quadratic problems
- MSK_IPAR_CHECK_TASK_DATA 190
  If this feature is turned on, then the task data is checked for bad values i.e. NaNs. before an optimization is performed.
- MSK_DPAR_DATA_TOL_AIJ 161
  Data tolerance threshold.
- MSK_DPAR_DATA_TOL_AIJ_LARGE 161
  Data tolerance threshold.
- MSK_DPAR_DATA_TOL_BOUND_INF 162
  Data tolerance threshold.
- MSK_DPAR_DATA_TOL_BOUND_WRN 162
  Data tolerance threshold.
- MSK_DPAR_DATA_TOL_C_HUGE 162
  Data tolerance threshold.
- MSK_DPAR_DATA_TOL_CJ_LARGE 162
  Data tolerance threshold.
- MSK_DPAR_DATA_TOL_QIJ 163
  Data tolerance threshold.
- MSK_DPAR_DATA_TOL_X 163
  Data tolerance threshold.

H.2 Double parameters

- MSK_DPAR_BASIS_REL_TOL_S 160
  Maximum relative dual bound violation allowed in an optimal basic solution.
- MSK_DPAR_BASIS_TOL_S 160
  Maximum absolute dual bound violation in an optimal basic solution.
- MSK_DPAR_BASIS_TOL_X 160
  Maximum absolute primal bound violation allowed in an optimal basic solution.
- MSK_DPAR_BI_LU_TOL_REL_PIV 161
  Relative pivot tolerance used in the LU factorization in the basis identification procedure.

- MSK_DPAR_CALLBACK_FREQ . . . 161
  Controls progress call-back frequency.

- MSK_DPAR_DATA_TOL_AIJ . . . 161
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_AIJ_LARGE . . . 161
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_BOUND_INF . . . 162
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_BOUND_WRN . . . 162
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_C_HUGE . . . 162
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_CJ_LARGE . . . 162
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_QIJ . . . 163
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_X . . . 163
  Data tolerance threshold.

- MSK_DPAR_FEASREPAIR_TOL . . . 163
  Tolerance for constraint enforcing upper bound on sum of weighted violations in feasibility repair.

- MSK_DPAR_INTPNT_CO_TOL_DFEAS . . . 163
  Dual feasibility tolerance used by the conic interior-point optimizer.

- MSK_DPAR_INTPNT_CO_TOL_INFEAS . . . 164
  Infeasibility tolerance for the conic solver.

- MSK_DPAR_INTPNT_CO_TOL_MU_RED . . . 164
  Optimality tolerance for the conic solver.

- MSK_DPAR_INTPNT_CO_TOL_NEAR_REL . . . 164
  Optimality tolerance for the conic solver.

- MSK_DPAR_INTPNT_CO_TOL_PFEAS . . . 164
  Primal feasibility tolerance used by the conic interior-point optimizer.

- MSK_DPAR_INTPNT_CO_TOL_REL_GAP . . . 165
  Relative gap termination tolerance used by the conic interior-point optimizer.

- MSK_DPAR_INTPNT_NL_MERIT_BAL . . . 165
  Controls if the complementarity and infeasibility is converging to zero at about equal rates.

- MSK_DPAR_INTPNT_NL_TOL_DFEAS . . . 165
  Dual feasibility tolerance used when a nonlinear model is solved.

- `MSK_DPAR_INTPNT_NL_TOL_MU_RED` 165
  Relative complementarity gap tolerance.
- `MSK_DPAR_INTPNT_NL_TOL_NEAR_REL` 166
  Nonlinear solver optimality tolerance parameter.
- `MSK_DPAR_INTPNT_NL_TOL_PFEAS` 166
  Primal feasibility tolerance used when a nonlinear model is solved.
- `MSK_DPAR_INTPNT_NL_TOL_REL_GAP` 166
  Relative gap termination tolerance for nonlinear problems.
- `MSK_DPAR_INTPNT_NL_TOL_REL_STEP` 166
  Relative step size to the boundary for general nonlinear optimization problems.
- `MSK_DPAR_INTPNT_TOL_DFEAS` 167
  Dual feasibility tolerance used for linear and quadratic optimization problems.
- `MSK_DPAR_INTPNT_TOL_DSAFE` 167
  Controls the interior-point dual starting point.
- `MSK_DPAR_INTPNT_TOL_INFEAS` 167
  Nonlinear solver infeasibility tolerance parameter.
- `MSK_DPAR_INTPNT_TOL_MU_RED` 167
  Relative complementarity gap tolerance.
- `MSK_DPAR_INTPNT_TOL_PATH` 168
  interior-point centering aggressiveness.
- `MSK_DPAR_INTPNT_TOL_PFEAS` 168
  Primal feasibility tolerance used for linear and quadratic optimization problems.
- `MSK_DPAR_INTPNT_TOL_PSAFE` 168
  Controls the interior-point primal starting point.
- `MSK_DPAR_INTPNT_TOL_REL_GAP` 168
  Relative gap termination tolerance.
- `MSK_DPAR_INTPNT_TOL_REL_STEP` 169
  Relative step size to the boundary for linear and quadratic optimization problems.
- `MSK_DPAR_INTPNT_TOL_STEP_SIZE` 169
  If the step size falls below the value of this parameter, then the interior-point optimizer assumes it is stalled. It it does not not make any progress.
- `MSK_DPAR_LOWER_OBJ_CUT` 169
  Objective bound.
- `MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH` 169
  Objective bound.

- `MSK_DPAR_MIO_DISABLE_TERM_TIME` 170
  Certain termination criterias is disabled within the mixed integer optimizer for period time specified by the parameter.
- `MSK_DPAR_MIO_HEURISTIC_TIME` 170
  Time limit for the mixed integer heuristics.
- `MSK_DPAR_MIO_MAX_TIME` 171
  Time limit for the mixed integer optimizer.
- `MSK_DPAR_MIO_MAX_TIME_APRX_OPT` 171
  Time limit for the mixed integer optimizer.
- `MSK_DPAR_MIO_NEAR_TOL_ABS_GAP` 171
  Relaxed absolute optimality tolerance employed by the mixed integer optimizer.
- `MSK_DPAR_MIO_NEAR_TOL_REL_GAP` 171
  The mixed integer optimizer is terminated when this tolerance is satisfied.
- `MSK_DPAR_MIO_REL_ADD_CUT_LIMITED` 172
  Controls cut generation for MIP solver.
- `MSK_DPAR_MIO_TOL_ABS_GAP` 172
  Absolute optimality tolerance employed by the mixed integer optimizer.
- `MSK_DPAR_MIO_TOL_ABS_RELAX_INT` 172
  Integer constraint tolerance.
- `MSK_DPAR_MIO_TOL_REL_GAP` 173
  Relative optimality tolerance employed by the mixed integer optimizer.
- `MSK_DPAR_MIO_TOL_REL_RELAX_INT` 173
  Integer constraint tolerance.
- `MSK_DPAR_MIO_TOL_X` 173
  Absolute solution tolerance used in mixed-integer optimizer.
- `MSK_DPAR_NONCONVEX_TOL_FEAS` 173
  Feasibility tolerance used by the nonconvex optimizer.
- `MSK_DPAR_NONCONVEX_TOL_OPT` 174
  Optimality tolerance used by the nonconvex optimizer.
- `MSK_DPAR_OPTIMIZER_MAX_TIME` 174
  Solver time limit.
- `MSK_DPAR_PRESOLVE_TOL_AIJ` 174
  Absolute zero tolerance employed for constraint coefficients in the presolve.
- `MSK_DPAR_PRESOLVE_TOL_LIN_DEP` 174
  Controls when a constraint is determined to be linearly dependent.

- `basis_rel_tol_s`

  Corresponding constant:
  `MSK_DPAR_BASIS_REL_TOL_S`

  Description:
  Maximum relative dual bound violation allowed in an optimal basic solution.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-12

- `basis_tol_s`

  Corresponding constant:
  `MSK_DPAR_BASIS_TOL_S`

  Description:
  Maximum absolute dual bound violation in an optimal basic solution.

  Possible Values:
  Any number between 1.0e-9 and +inf.

  Default value:
  1.0e-6

- `basis_tol_x`

  Corresponding constant:
  `MSK_DPAR_BASIS_TOL_X`

  Description:
  Maximum absolute primal bound violation allowed in an optimal basic solution.

  Possible Values:
  Any number between 1.0e-9 and +inf.

  Default value:
  1.0e-6

- `bi_lu_tol_rel_piv`

  Corresponding constant:
  `MSK_DPAR_BI_LU_TOL_REL_PIV`

  Description:
  Relative pivot tolerance used in the LU factorization in the basis identification procedure.

  Possible Values:
  Any number between 1.0e-6 and 0.999999.

  Default value:
  0.01

- `callback_freq`

  Corresponding constant:
  `MSK_DPAR_CALLBACK_FREQ`

  Description:
  Controls the time between calls to the progress call-back function. Hence, if the value of this parameter is for example 10, then the call-back is called approximately each 10 seconds. A negative value is equivalent to infinity.

  In general frequent call-backs may hurt the performance.

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1.0

- `data_tol_aij`

  Corresponding constant:
  `MSK_DPAR_DATA_TOL_AIJ`

  Description:
  Absolute zero tolerance for elements in A.

  Possible Values:
  Any number between 1.0e-16 and 1.0e-6.

  Default value:
  1.0e-12

- `data_tol_aij_large`

  Corresponding constant:
  `MSK_DPAR_DATA_TOL_AIJ_LARGE`

  Description:
  An element in A which is larger than this value in absolute size causes a warning message to be printed.

  Possible Values:
  Any number between 0.0 and +inf.

Default value:
1.0e10

- `data_tol_bound_inf`

  Corresponding constant:
  `MSK_DPAR_DATA_TOL_BOUND_INF`

  Description:
  Any bound which in absolute value is greater than this parameter is considered infinite.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e16

- `data_tol_bound_wrn`

  Corresponding constant:
  `MSK_DPAR_DATA_TOL_BOUND_WRN`

  Description:
  If a bound value is larger than this value in absolute size, then a warning message is issued.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e8

- `data_tol_c_huge`

  Corresponding constant:
  `MSK_DPAR_DATA_TOL_C_HUGE`

  Description:
  An element in c which is larger than the value of this parameter in absolute terms is considered to be huge and generates an error.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e16

- `data_tol_cj_large`

  Corresponding constant:
  `MSK_DPAR_DATA_TOL_CJ_LARGE`

  Description:
  An element in c which is larger than this value in absolute terms causes a warning message to be printed.

  Possible Values:
  Any number between 0.0 and +inf.

Default value:
1.0e8

- `data_tol_qij`

  Corresponding constant:
  `MSK_DPAR_DATA_TOL_QIJ`

  Description:
  Absolute zero tolerance for elements in Q matrices.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-16

- `data_tol_x`

  Corresponding constant:
  `MSK_DPAR_DATA_TOL_X`

  Description:
  Zero tolerance for constraints and variables i.e. if the distance between the lower and upper bound is less than this value, then the lower and lower bound is considered identical.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-8

- `feasrepair_tol`

  Corresponding constant:
  `MSK_DPAR_FEASREPAIR_TOL`

  Description:
  Tolerance for constraint enforcing upper bound on sum of weighted violations in feasibility repair.

  Possible Values:
  Any number between 1.0e-16 and 1.0e+16.

  Default value:
  1.0e-10

- `intpnt_co_tol_dfeas`

  Corresponding constant:
  `MSK_DPAR_INTPNT_CO_TOL_DFEAS`

  Description:
  Dual feasibility tolerance used by the conic interior-point optimizer.

  Possible Values:
  Any number between 0.0 and 1.0.

Default value:
1.0e-8

- `intpnt_co_tol_infeas`

  Corresponding constant:
  `MSK_DPAR_INTPNT_CO_TOL_INFEAS`

  Description:
  Controls when the conic interior-point optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-8

- `intpnt_co_tol_mu_red`

  Corresponding constant:
  `MSK_DPAR_INTPNT_CO_TOL_MU_RED`

  Description:
  Relative complementarity gap tolerance feasibility tolerance used by the conic interior-point optimizer.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-8

- `intpnt_co_tol_near_rel`

  Corresponding constant:
  `MSK_DPAR_INTPNT_CO_TOL_NEAR_REL`

  Description:
  If MOSEK cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.

  Possible Values:
  Any number between 1.0 and +inf.

  Default value:
  100

- `intpnt_co_tol_pfeas`

  Corresponding constant:
  `MSK_DPAR_INTPNT_CO_TOL_PFEAS`

Description:
Primal feasibility tolerance used by the conic interior-point optimizer.

Possible Values:
Any number between 0.0 and 1.0.

Default value:
1.0e-8

- `intpnt_co_tol_rel_gap`

  Corresponding constant:
  `MSK_DPAR_INTPNT_CO_TOL_REL_GAP`

  Description:
  Relative gap termination tolerance used by the conic interior-point optimizer.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-8

- `intpnt_nl_merit_bal`

  Corresponding constant:
  `MSK_DPAR_INTPNT_NL_MERIT_BAL`

  Description:
  Controls if the complementarity and infeasibility is converging to zero at about equal rates.

  Possible Values:
  Any number between 0.0 and 0.99.

  Default value:
  1.0e-4

- `intpnt_nl_tol_dfeas`

  Corresponding constant:
  `MSK_DPAR_INTPNT_NL_TOL_DFEAS`

  Description:
  Dual feasibility tolerance used when a nonlinear model is solved.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-8

- `intpnt_nl_tol_mu_red`

  Corresponding constant:
  `MSK_DPAR_INTPNT_NL_TOL_MU_RED`

  Description:
  Relative complementarity gap tolerance.

Possible Values:
Any number between 0.0 and 1.0.

Default value:
1.0e-12

- `intpnt_nl_tol_near_rel`

  Corresponding constant:
  `MSK_DPAR_INTPNT_NL_TOL_NEAR_REL`

  Description:
  If the MOSEK nonlinear interior-point optimizer cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.

  Possible Values:
  Any number between 1.0 and +inf.

  Default value:
  1000.0

- `intpnt_nl_tol_pfeas`

  Corresponding constant:
  `MSK_DPAR_INTPNT_NL_TOL_PFEAS`

  Description:
  Primal feasibility tolerance used when a nonlinear model is solved.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-8

- `intpnt_nl_tol_rel_gap`

  Corresponding constant:
  `MSK_DPAR_INTPNT_NL_TOL_REL_GAP`

  Description:
  Relative gap termination tolerance for nonlinear problems.

  Possible Values:
  Any number between 1.0e-14 and +inf.

  Default value:
  1.0e-6

- `intpnt_nl_tol_rel_step`

  Corresponding constant:
  `MSK_DPAR_INTPNT_NL_TOL_REL_STEP`

Description:
Relative step size to the boundary for general nonlinear optimization problems.

Possible Values:
Any number between 1.0e-4 and 0.9999999.

Default value:
0.995

- `intpnt_tol_dfeas`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_DFEAS`

  Description:
  Dual feasibility tolerance used for linear and quadratic optimization problems.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-8

- `intpnt_tol_dsafe`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_DSAFE`

  Description:
  Controls the initial dual starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly.

  Possible Values:
  Any number between 1.0e-4 and +inf.

  Default value:
  1.0

- `intpnt_tol_infeas`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_INFEAS`

  Description:
  Controls when the optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-8

- `intpnt_tol_mu_red`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_MU_RED`

Description:
Relative complementarity gap tolerance.

Possible Values:
Any number between 0.0 and 1.0.

Default value:
1.0e-16

- `intpnt_tol_path`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_PATH`

  Description:
  Controls how close the interior-point optimizer follows the central path. A large value of this parameter means the central is followed very closely. On numerical unstable problems it might worthwhile to increase this parameter.

  Possible Values:
  Any number between 0.0 and 0.9999.

  Default value:
  1.0e-8

- `intpnt_tol_pfeas`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_PFEAS`

  Description:
  Primal feasibility tolerance used for linear and quadratic optimization problems.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-8

- `intpnt_tol_psafe`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_PSAFE`

  Description:
  Controls the initial primal starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly and/or the constraint or variable bounds are very large, then it might be worthwhile to increase this value.

  Possible Values:
  Any number between 1.0e-4 and +inf.

  Default value:
  1.0

- `intpnt_tol_rel_gap`

Corresponding constant:
`MSK_DPAR_INTPNT_TOL_REL_GAP`

Description:
Relative gap termination tolerance.

Possible Values:
Any number between 1.0e-14 and +inf.

Default value:
1.0e-8

- `intpnt_tol_rel_step`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_REL_STEP`

  Description:
  Relative step size to the boundary for linear and quadratic optimization problems.

  Possible Values:
  Any number between 1.0e-4 and 0.999999.

  Default value:
  0.9999

- `intpnt_tol_step_size`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_STEP_SIZE`

  Description:
  If the step size falls below the value of this parameter, then the interior-point optimizer assumes it is stalled. It it does not not make any progress.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-10

- `lower_obj_cut`

  Corresponding constant:
  `MSK_DPAR_LOWER_OBJ_CUT`

  Description:
  If a feasible solution having an objective value outside, the interval [`MSK_DPAR_LOWER_OBJ_CUT`, `MSK_DPAR_UPPER_OBJ_CUT`], then MOSEK is terminated.

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1.0e30

- `lower_obj_cut_finite_trh`

Corresponding constant:
`MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH`

Description:
If the lower objective cut is less than the value of this parameter value, then the lower objective cut i.e. `MSK_DPAR_LOWER_OBJ_CUT` is treated as $-\infty$.

Possible Values:
Any number between -inf and +inf.

Default value:
-0.5e30

- `mio_disable_term_time`

Corresponding constant:
`MSK_DPAR_MIO_DISABLE_TERM_TIME`

Description:
The termination criteria governed by

- `MSK_IPAR_MIO_MAX_NUM_RELAXS`
- `MSK_IPAR_MIO_MAX_NUM_BRANCHES`
- `MSK_DPAR_MIO_NEAR_TOL_ABS_GAP`
- `MSK_DPAR_MIO_NEAR_TOL_REL_GAP`

is disabled the first n seconds. This parameter specifies the number n. A negative value is identical to infinity i.e. the termination criterias are never checked.

Possible Values:
Any number between 0.0 and +inf.

Default value:
0.0

See also:

`MSK_IPAR_MIO_MAX_NUM_RELAXS` Maximum number of relaxations in branch and bound search.

`MSK_IPAR_MIO_MAX_NUM_BRANCHES` Maximum number of branches allowed during the branch and bound search.

`MSK_DPAR_MIO_NEAR_TOL_ABS_GAP` Relaxed absolute optimality tolerance employed by the mixed integer optimizer.

`MSK_DPAR_MIO_NEAR_TOL_REL_GAP` The mixed integer optimizer is terminated when this tolerance is satisfied.

- `mio_heuristic_time`

Corresponding constant:
`MSK_DPAR_MIO_HEURISTIC_TIME`

Description:
Minimum amount of time to be used in the heuristic search for a good feasible integer solution. A negative values implies that the optimizer decides the amount of time to be spent in the heuristic.

Possible Values:
Any number between -inf and +inf.

Default value:
-1.0

- `mio_max_time`

  Corresponding constant:
  `MSK_DPAR_MIO_MAX_TIME`

  Description:
  This parameter limits the maximum time spent by the mixed integer optimizer. A negative number means infinity.

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1.0

- `mio_max_time_aprx_opt`

  Corresponding constant:
  `MSK_DPAR_MIO_MAX_TIME_APRX_OPT`

  Description:
  Number of seconds spent by the mixed integer optimizer before the `MSK_DPAR_MIO_TOL_REL_RELAX_INT` is applied.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  60

- `mio_near_tol_abs_gap`

  Corresponding constant:
  `MSK_DPAR_MIO_NEAR_TOL_ABS_GAP`

  Description:
  Relaxed absolute optimality tolerance employed by the mixed integer optimizer. This termination criteria is delayed. See `MSK_DPAR_MIO_DISABLE_TERM_TIME` for details.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  0.0

  See also:

  `MSK_DPAR_MIO_DISABLE_TERM_TIME` Certain termination criterias is disabled within the mixed integer optimizer for period time specified by the parameter.

- `mio_near_tol_rel_gap`

Corresponding constant:
`MSK_DPAR_MIO_NEAR_TOL_REL_GAP`

Description:
The mixed integer optimizer is terminated when this tolerance is satisfied. This termination criteria is delayed. See `MSK_DPAR_MIO_DISABLE_TERM_TIME` for details.

Possible Values:
Any number between 0.0 and +inf.

Default value:
1.0e-5

See also:

`MSK_DPAR_MIO_DISABLE_TERM_TIME` Certain termination criterias is disabled within the mixed integer optimizer for period time specified by the parameter.

- `mio_rel_add_cut_limited`

Corresponding constant:
`MSK_DPAR_MIO_REL_ADD_CUT_LIMITED`

Description:
Controls how many cuts the mixed integer optimizer is allowed to add to the problem. Let α be the value of this parameter and m the number constraints, then mixed integer optimizer is allowed to αm cuts.

Possible Values:
Any number between 0.0 and 2.0.

Default value:
0.75

- `mio_tol_abs_gap`

Corresponding constant:
`MSK_DPAR_MIO_TOL_ABS_GAP`

Description:
Absolute optimality tolerance employed by the mixed integer optimizer.

Possible Values:
Any number between 0.0 and +inf.

Default value:
0.0

- `mio_tol_abs_relax_int`

Corresponding constant:
`MSK_DPAR_MIO_TOL_ABS_RELAX_INT`

Description:
Absolute relaxation tolerance of the integer constraints. I.e. $\min(|x| - \lfloor x \rfloor, \lceil x \rceil - |x|)$ is less than the tolerance then the integer restrictions assumed to be satisfied.

Possible Values:
Any number between 0.0 and +inf.

Default value:
1.0e-5

- `mio_tol_rel_gap`

  Corresponding constant:
  `MSK_DPAR_MIO_TOL_REL_GAP`

  Description:
  Relative optimality tolerance employed by the mixed integer optimizer.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-8

- `mio_tol_rel_relax_int`

  Corresponding constant:
  `MSK_DPAR_MIO_TOL_REL_RELAX_INT`

  Description:
  Relative relaxation tolerance of the integer constraints. I.e. $\min(|x| - \lfloor x \rfloor, \lceil x \rceil - |x|)$ is less than the tolerance times $|x|$ then the integer restrictions assumed to be satisfied.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-6

- `mio_tol_x`

  Corresponding constant:
  `MSK_DPAR_MIO_TOL_X`

  Description:
  Absolute solution tolerance used in mixed-integer optimizer.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-6

- `nonconvex_tol_feas`

  Corresponding constant:
  `MSK_DPAR_NONCONVEX_TOL_FEAS`

  Description:
  Feasibility tolerance used by the nonconvex optimizer.

Possible Values:
Any number between 0.0 and +inf.

Default value:
1.0e-6

- `nonconvex_tol_opt`

  Corresponding constant:
  `MSK_DPAR_NONCONVEX_TOL_OPT`

  Description:
  Optimality tolerance used by the nonconvex optimizer.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-7

- `optimizer_max_time`

  Corresponding constant:
  `MSK_DPAR_OPTIMIZER_MAX_TIME`

  Description:
  Maximum amount of time the optimizer is allowed to spent on the optimization. A negative number means infinity.

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1.0

- `presolve_tol_aij`

  Corresponding constant:
  `MSK_DPAR_PRESOLVE_TOL_AIJ`

  Description:
  Absolute zero tolerance employed for a_{ij} in the presolve.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-12

- `presolve_tol_lin_dep`

  Corresponding constant:
  `MSK_DPAR_PRESOLVE_TOL_LIN_DEP`

  Description:
  Controls when a constraint is determined to be linearly dependent.

Possible Values:
Any number between 0.0 and +inf.

Default value:
1.0e-6

- `presolve_tol_s`

  Corresponding constant:
  `MSK_DPAR_PRESOLVE_TOL_S`

  Description:
  Absolute zero tolerance employed for s_i in the presolve.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-8

- `presolve_tol_x`

  Corresponding constant:
  `MSK_DPAR_PRESOLVE_TOL_X`

  Description:
  Absolute zero tolerance employed for x_j in the presolve.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-8

- `simplex_abs_tol_piv`

  Corresponding constant:
  `MSK_DPAR_SIMPLEX_ABS_TOL_PIV`

  Description:
  Absolute pivot tolerance employed by the simplex optimizers.

  Possible Values:
  Any number between 1.0e-12 and +inf.

  Default value:
  1.0e-7

- `upper_obj_cut`

  Corresponding constant:
  `MSK_DPAR_UPPER_OBJ_CUT`

  Description:
  If a feasible solution having and objective value outside, the interval [`MSK_DPAR_LOWER_OBJ_CUT`, `MSK_DPAR_UPPER_OBJ_CUT`], then MOSEK is terminated.

Possible Values:
Any number between -inf and +inf.

Default value:
1.0e30

- `upper_obj_cut_finite_trh`

  Corresponding constant:
  `MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH`

  Description:
  If the upper objective cut is greater than the value of this value parameter, then the the upper objective cut `MSK_DPAR_UPPER_OBJ_CUT` is treated as ∞.

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  0.5e30

H.3 Integer parameters

- `MSK_IPAR_ALLOC_ADD_QNZ` 187
  Controls how the quadratic matrixes are extended.

- `MSK_IPAR_BI_CLEAN_OPTIMIZER` 187
  Controls which simplex optimizer is used in the clean-up phase.

- `MSK_IPAR_BI_IGNORE_MAX_ITER` 188
  Turns on basis identification in case the interior-point optimizer is terminated due to maximum number of iterations.

- `MSK_IPAR_BI_IGNORE_NUM_ERROR` 188
  Turns on basis identification in case the interior-point optimizer is terminated due to a numerical problem.

- `MSK_IPAR_BI_MAX_ITERATIONS` 188
  Maximum number of iterations after basis identification.

- `MSK_IPAR_CACHE_SIZE_L1` 189
  Specifies the size of the level 1 cache of the processor.

- `MSK_IPAR_CACHE_SIZE_L2` 189
  Specifies the size of the level 2 cache of the processor.

- `MSK_IPAR_CHECK_CONVEXITY` 189
  Specify the level of convexity check on quadratic problems

- `MSK_IPAR_CHECK_CTRL_C` 190
  Turns ctrl-c check on or off.

- `MSK_IPAR_CHECK_TASK_DATA` 190
  If this feature is turned on, then the task data is checked for bad values i.e. NaNs. before an optimization is performed.

- `MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS` 190
  The maximum number of simultaneous optimizations that will be started by the concurrent optimizer.

- `MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX` 191
  Priority of the dual simplex algorithm when selecting solvers for concurrent optimization.

- `MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX` 191
  Priority of the free simplex optimizer when selecting solvers for concurrent optimization.

- `MSK_IPAR_CONCURRENT_PRIORITY_INTPNT` 191
  Priority of the interior-point algorithm when selecting solvers for concurrent optimization.

- `MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX` 191
  Priority of the primal simplex algorithm when selecting solvers for concurrent optimization.

- `MSK_IPAR_CPU_TYPE` 191
  Specifies the CPU type.

- `MSK_IPAR_DATA_CHECK` 192
  Enable data checking for debug purposes.

- `MSK_IPAR_FEASREPAIR_OPTIMIZE` 192
  Controls which type of feasibility analysis is to be performed.

- `MSK_IPAR_FLUSH_STREAM_FREQ` 193
  Controls the stream flushing frequency.

- `MSK_IPAR_INFEAS_GENERIC_NAMES` 193
  Controls the contents of the infeasibility report.

- `MSK_IPAR_INFEAS_PREFER_PRIMAL` 193
  Controls which certificate is used if both primal- and dual- certificate of infeasibility is available.

- `MSK_IPAR_INFEAS_REPORT_AUTO` 193
  Turns the feasibility report on or off.

- `MSK_IPAR_INFEAS_REPORT_LEVEL` 194
  Controls the contents of the infeasibility report.

- `MSK_IPAR_INTPNT_BASIS` 194
  Controls whether basis identification is performed.

- `MSK_IPAR_INTPNT_DIFF_STEP` 195
  Controls whether different step sizes are allowed in the primal and dual space.

- `MSK_IPAR_INTPNT_FACTOR_DEBUG_LVL` 195
  Controls factorization debug level.

- `MSK_IPAR_INTPNT_FACTOR_METHOD` 195
  Controls the method used to factor the Newton equation system.
- `MSK_IPAR_INTPNT_MAX_ITERATIONS` 195
  Controls the maximum number of iterations allowed in the interior-point optimizer.
- `MSK_IPAR_INTPNT_MAX_NUM_COR` 196
  Maximum number of correction steps.
- `MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS` 196
  Maximum number of steps to be used by the iterative search direction refinement.
- `MSK_IPAR_INTPNT_NUM_THREADS` 196
  Controls the number of threads employed by the interior-point optimizer.
- `MSK_IPAR_INTPNT_OFF_COL_TRH` 196
  Controls the aggressiveness of the offending column detection.
- `MSK_IPAR_INTPNT_ORDER_METHOD` 197
  Controls the ordering strategy.
- `MSK_IPAR_INTPNT_REGULARIZATION_USE` 197
  Controls whether regularization is allowed.
- `MSK_IPAR_INTPNT_SCALING` 197
  Controls how the problem is scaled before the interior-point optimizer is used.
- `MSK_IPAR_INTPNT_SOLVE_FORM` 198
  Controls whether the primal or the dual problem is solved.
- `MSK_IPAR_INTPNT_STARTING_POINT` 198
  Starting point used by the interior-point optimizer.
- `MSK_IPAR_LICENSE_ALLOW_OVERUSE` 198
  Controls if license overuse is allowed when caching licenses
- `MSK_IPAR_LICENSE_CACHE_TIME` 199
  Controls the license manager client behavior.
- `MSK_IPAR_LICENSE_CHECK_TIME` 199
  Controls the license manager client behavior.
- `MSK_IPAR_LICENSE_DEBUG` 199
  Controls the license manager client debugging behavior.
- `MSK_IPAR_LICENSE_PAUSE_TIME` 199
  Controls license manager client behavior.
- `MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS` 200
  Controls license manager client behavior.
- `MSK_IPAR_LICENSE_WAIT` 200
  Controls if MOSEK should queue for a license if none is available.

- MSK_IPAR_LOG 200
  Controls the amount of log information.

- MSK_IPAR_LOG_BI 201
  Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.

- MSK_IPAR_LOG_BI_FREQ 201
  Controls the logging frequency.

- MSK_IPAR_LOG_CONCURRENT 201
  Controls amount of output printed by the concurrent optimizer.

- MSK_IPAR_LOG_CUT_SECOND_OPT 201
  Controls the reduction in the log levels for the second and any subsequent optimizations.

- MSK_IPAR_LOG_FACTOR 202
  If turned on, then the factor log lines are added to the log.

- MSK_IPAR_LOG_FEASREPAIR 202
  Controls the amount of output printed when performing feasibility repair.

- MSK_IPAR_LOG_FILE 202
  If turned on, then some log info is printed when a file is written or read.

- MSK_IPAR_LOG_HEAD 203
  If turned on, then a header line is added to the log.

- MSK_IPAR_LOG_INFEAS_ANA 203
  Controls log level for the infeasibility analyzer.

- MSK_IPAR_LOG_INTPNT 203
  Controls the amount of log information from the interior-point optimizers.

- MSK_IPAR_LOG_MIO 203
  Controls the amount of log information from the mixed-integer optimizers.

- MSK_IPAR_LOG_MIO_FREQ 204
  The mixed integer solver logging frequency.

- MSK_IPAR_LOG_NONCONVEX 204
  Controls amount of output printed by the nonconvex optimizer.

- MSK_IPAR_LOG_OPTIMIZER 204
  Controls the amount of general optimizer information that is logged.

- MSK_IPAR_LOG_ORDER 204
  If turned on, then factor lines are added to the log.

- MSK_IPAR_LOG_PARAM 205
  Controls the amount of information printed out about parameter changes.

- `MSK_IPAR_LOG_PRESOLVE` 205
  Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.
- `MSK_IPAR_LOG_RESPONSE` 205
  Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.
- `MSK_IPAR_LOG_SENSITIVITY` 205
  Control logging in sensitivity analyzer.
- `MSK_IPAR_LOG_SENSITIVITY_OPT` 206
  Control logging in sensitivity analyzer.
- `MSK_IPAR_LOG_SIM` 206
  Controls the amount of log information from the simplex optimizers.
- `MSK_IPAR_LOG_SIM_FREQ` 206
  Controls simplex logging frequency.
- `MSK_IPAR_LOG_SIM_MINOR` 206
  Currently not in use.
- `MSK_IPAR_LOG_SIM_NETWORK_FREQ` 207
  Controls the network simplex logging frequency.
- `MSK_IPAR_LOG_STORAGE` 207
  Controls the memory related log information.
- `MSK_IPAR_LP_WRITE_IGNORE_INCOMPATIBLE_ITEMS` 207
  Controls the result of writing a problem containing incompatible items to an LP file.
- `MSK_IPAR_MAX_NUM_WARNINGS` 208
  Waning level. A higher value results in more warnings.
- `MSK_IPAR_MAXNUMANZ_DOUBLE_TRH` 208
  Controls how the constraint matrix is extended.
- `MSK_IPAR_MIO_BRANCH_DIR` 208
  Controls whether the mixed integer optimizer is branching up or down by default.
- `MSK_IPAR_MIO_BRANCH_PRIORITIES_USE` 208
  Controls whether branching priorities are used by the mixed integer optimizer.
- `MSK_IPAR_MIO_CONSTRUCT_SOL` 209
  Controls if initial MIP solution should be constructed from value of integer variables.
- `MSK_IPAR_MIO_CONT_SOL` 209
  Controls the meaning of interior-point and basic solutions in MIP problems.
- `MSK_IPAR_MIO_CUT_LEVEL_ROOT` 209
  Controls the cut level employed by the mixed integer optimizer at the root node.

- `MSK_IPAR_MIO_CUT_LEVEL_TREE` 210
  Controls the cut level employed by the mixed integer optimizer in the tree.

- `MSK_IPAR_MIO_FEASPUMP_LEVEL` 210
  Controls the feasibility pump heuristic which is used to construct a good initial feasible solution.

- `MSK_IPAR_MIO_HEURISTIC_LEVEL` 211
  Controls the heuristic employed by the mixed integer optimizer to locate an initial integer feasible solution.

- `MSK_IPAR_MIO_KEEP_BASIS` 211
  Controls whether the integer presolve keeps bases in memory.

- `MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER` 211
  Controls the size of the local search space when doing local branching.

- `MSK_IPAR_MIO_MAX_NUM_BRANCHES` 211
  Maximum number of branches allowed during the branch and bound search.

- `MSK_IPAR_MIO_MAX_NUM_RELAXS` 212
  Maximum number of relaxations in branch and bound search.

- `MSK_IPAR_MIO_MAX_NUM_SOLUTIONS` 212
  Controls how many feasible solutions the mixed-integer optimizer investigates.

- `MSK_IPAR_MIO_MODE` 212
  Turns on/off the mixed integer mode.

- `MSK_IPAR_MIO_NODE_OPTIMIZER` 213
  Controls which optimizer is employed at the non-root nodes in the mixed integer optimizer.

- `MSK_IPAR_MIO_NODE_SELECTION` 213
  Controls the node selection strategy employed by the mixed integer optimizer.

- `MSK_IPAR_MIO_PRESOLVE_AGGREGATE` 214
  Controls whether problem aggregation is performed in the mixed integer presolve.

- `MSK_IPAR_MIO_PRESOLVE_PROBING` 214
  Controls whether probing is employed by the mixed integer presolve.

- `MSK_IPAR_MIO_PRESOLVE_USE` 214
  Controls whether presolve is performed by the mixed integer optimizer.

- `MSK_IPAR_MIO_ROOT_OPTIMIZER` 215
  Controls which optimizer is employed at the root node in the mixed integer optimizer.

- `MSK_IPAR_MIO_STRONG_BRANCH` 215
  The depth from the root in which strong branching is employed.

- `MSK_IPAR_NONCONVEX_MAX_ITERATIONS` 215
  Maximum number of iterations that can be used by the nonconvex optimizer.

- MSK_IPAR_OBJECTIVE_SENSE 216
  If the objective sense for the task is undefined, then the value of this parameter is used as the default objective sense.

- MSK_IPAR_OPF_MAX_TERMS_PER_LINE 216
  The maximum number of terms (linear and quadratic) per line when an OPF file is written.

- MSK_IPAR_OPF_WRITE_HEADER 216
  Write a text header with date and MOSEK version in an OPF file.

- MSK_IPAR_OPF_WRITE_HINTS 216
  Write a hint section with problem dimensions in the beginning of an OPF file.

- MSK_IPAR_OPF_WRITE_PARAMETERS 217
  Write a parameter section in an OPF file.

- MSK_IPAR_OPF_WRITE_PROBLEM 217
  Write objective, constraints, bounds etc. to an OPF file.

- MSK_IPAR_OPF_WRITE_SOL_BAS 217
  Controls what is written to the OPF files.

- MSK_IPAR_OPF_WRITE_SOL_ITG 218
  Controls what is written to the OPF files.

- MSK_IPAR_OPF_WRITE_SOL_ITR 218
  Controls what is written to the OPF files.

- MSK_IPAR_OPF_WRITE_SOLUTIONS 218
  Enable inclusion of solutions in the OPF files.

- MSK_IPAR_OPTIMIZER 218
  Controls which optimizer is used to optimize the task.

- MSK_IPAR_PARAM_READ_CASE_NAME 219
  If turned on, then names in the parameter file are case sensitive.

- MSK_IPAR_PARAM_READ_IGN_ERROR 219
  If turned on, then errors in paramter settings is ignored.

- MSK_IPAR_PRESOLVE_ELIM_FILL 220
  Maximum amount of fill-in in the elimination phase.

- MSK_IPAR_PRESOLVE_ELIMINATOR_USE 220
  Controls whether free or implied free variables are eliminated from the problem.

- MSK_IPAR_PRESOLVE_LEVEL 220
  Currently not used.

- MSK_IPAR_PRESOLVE_LINDEP_USE 220
  Controls whether the linear constraints are checked for linear dependencies.

- `MSK_IPAR_PRESOLVE_LINDEP_WORK_LIM` 221
  Controls linear dependency check in presolve.
- `MSK_IPAR_PRESOLVE_USE` 221
  Controls whether the presolve is applied to a problem before it is optimized.
- `MSK_IPAR_READ_ADD_ANZ` 221
  Controls how the constraint matrix is extended.
- `MSK_IPAR_READ_ADD_CON` 221
  Additional number of constraints that is made room for in the problem.
- `MSK_IPAR_READ_ADD_CONE` 222
  Additional number of conic constraints that is made room for in the problem.
- `MSK_IPAR_READ_ADD_QNZ` 222
  Controls how the quadratic matrixes are extended.
- `MSK_IPAR_READ_ADD_VAR` 222
  Additional number of variables that is made room for in the problem.
- `MSK_IPAR_READ_ANZ` 222
  Controls the expected number of constraint non-zeros.
- `MSK_IPAR_READ_CON` 223
  Controls the expected number of constraints.
- `MSK_IPAR_READ_CONE` 223
  Controls the expected number of conic constraints.
- `MSK_IPAR_READ_DATA_COMPRESSED` 223
  Controls the input file decompression.
- `MSK_IPAR_READ_DATA_FORMAT` 223
  Format of the data file to be read.
- `MSK_IPAR_READ_KEEP_FREE_CON` 224
  Controls whether the free constraints are included in the problem.
- `MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU` 224
  Controls how the LP files are interpreted.
- `MSK_IPAR_READ_LP_QUOTED_NAMES` 224
  If a name is in quotes when reading an LP file, the quotes will be removed.
- `MSK_IPAR_READ_MPS_FORMAT` 225
  Controls how strictly the MPS file reader interprets the MPS format.
- `MSK_IPAR_READ_MPS_KEEP_INT` 225
  Controls if integer constraints are read.
- `MSK_IPAR_READ_MPS_OBJ_SENSE` 225
  Controls the MPS format extensions.

- MSK_IPAR_READ_MPS_QUOTED_NAMES 226
  Controls the MPS format extensions.

- MSK_IPAR_READ_MPS_RELAX 226
  Controls the meaning of integer constraints.

- MSK_IPAR_READ_MPS_WIDTH 226
  Controls the maximal number of chars allowed in one line of the MPS file.

- MSK_IPAR_READ_Q_MODE 226
  Controls how the Q matrices are read from the MPS file.

- MSK_IPAR_READ_QNZ 227
  Controls the expected number of quadratic non-zeros.

- MSK_IPAR_READ_TASK_IGNORE_PARAM 227
  Controls what information is used from the task files.

- MSK_IPAR_READ_VAR 227
  Controls the expected number of variables.

- MSK_IPAR_SENSITIVITY_ALL 228
  Controls sensitivity report behavior.

- MSK_IPAR_SENSITIVITY_OPTIMIZER 228
  Controls which optimizer is used for optimal partition sensitivity analysis.

- MSK_IPAR_SENSITIVITY_TYPE 228
  Controls which type of sensitivity analysis is to be performed.

- MSK_IPAR_SIM_DEGEN 229
  Controls how aggressive degeneration is approached.

- MSK_IPAR_SIM_DUAL_CRASH 229
  Controls whether crashing is performed in the dual simplex optimizer.

- MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION 229
  Controls how aggressively restricted selection is used.

- MSK_IPAR_SIM_DUAL_SELECTION 230
  Controls the dual simplex strategy.

- MSK_IPAR_SIM_HOTSTART 230
  Controls the type of hot-start that the simplex optimizer perform.

- MSK_IPAR_SIM_MAX_ITERATIONS 231
  Maximum number of iterations that can be used by a simplex optimizer.

- MSK_IPAR_SIM_MAX_NUM_SETBACKS 231
  Controls how many setbacks that are allowed within a simplex optimizer.

- MSK_IPAR_SIM_NETWORK_DETECT 231
  Level of aggressiveness of network detection.

- MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART 232
  Level of aggressiveness of network detection in a simplex hot-start.
- MSK_IPAR_SIM_NETWORK_DETECT_METHOD 232
  Controls which type of detection method the network extraction should use.
- MSK_IPAR_SIM_NON_SINGULAR 232
  Controls if the simplex optimizer ensures a non-singular basis, if possible.
- MSK_IPAR_SIM_PRIMAL_CRASH 232
  Controls the simplex crash.
- MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION 233
  Controls how aggressively restricted selection is used.
- MSK_IPAR_SIM_PRIMAL_SELECTION 233
  Controls the primal simplex strategy.
- MSK_IPAR_SIM_REFACTOR_FREQ 234
  Controls the basis refactoring frequency.
- MSK_IPAR_SIM_SAVE_LU 234
  Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.
- MSK_IPAR_SIM_SCALING 234
  Controls how the problem is scaled before a simplex optimizer is used.
- MSK_IPAR_SIM_SOLVE_FORM 235
  Controls whether the primal or the dual problem is solved by the primal-/dual- simplex optimizer.
- MSK_IPAR_SIM_STABILITY_PRIORITY 235
  Controls how high priority the numerical stability should be given.
- MSK_IPAR_SIM_SWITCH_OPTIMIZER 235
  Controls the simplex behavior.
- MSK_IPAR_SOL_FILTER_KEEP_BASIC 235
  Controls the license manager client behavior.
- MSK_IPAR_SOL_FILTER_KEEP_RANGED 236
  Control the contents of the solution files.
- MSK_IPAR_SOL_QUOTED_NAMES 236
  Controls the solution file format.
- MSK_IPAR_SOL_READ_NAME_WIDTH 236
  Controls the input solution file format.
- MSK_IPAR_SOL_READ_WIDTH 237
  Controls the input solution file format.

- MSK_IPAR_SOLUTION_CALLBACK 237
  Indicates whether solution call-backs will be performed during the optimization.

- MSK_IPAR_WARNING_LEVEL 237
  Warning level.

- MSK_IPAR_WRITE_BAS_CONSTRAINTS 237
  Controls the basic solution file format.

- MSK_IPAR_WRITE_BAS_HEAD 238
  Controls the basic solution file format.

- MSK_IPAR_WRITE_BAS_VARIABLES 238
  Controls the basic solution file format.

- MSK_IPAR_WRITE_DATA_COMPRESSED 238
  Controls output file compression.

- MSK_IPAR_WRITE_DATA_FORMAT 239
  Controls the output file problem format.

- MSK_IPAR_WRITE_DATA_PARAM 239
  Controls output file data.

- MSK_IPAR_WRITE_FREE_CON 239
  Controls the output file data.

- MSK_IPAR_WRITE_GENERIC_NAMES 240
  Controls the output file data.

- MSK_IPAR_WRITE_GENERIC_NAMES_IO 240
  Index origin used in generic names.

- MSK_IPAR_WRITE_INT_CONSTRAINTS 240
  Controls the integer solution file format.

- MSK_IPAR_WRITE_INT_HEAD 240
  Controls the integer solution file format.

- MSK_IPAR_WRITE_INT_VARIABLES 241
  Controls the integer solution file format.

- MSK_IPAR_WRITE_LP_LINE_WIDTH 241
  Controls the LP output file format.

- MSK_IPAR_WRITE_LP_QUOTED_NAMES 241
  Controls LP output file format.

- MSK_IPAR_WRITE_LP_STRICT_FORMAT 241
  Controls whether LP output files satisfy the LP format strictly.

- MSK_IPAR_WRITE_LP_TERMS_PER_LINE 242
  Controls the LP output file format.

- `MSK_IPAR_WRITE_MPS_INT` 242
  Controls the output file data.

- `MSK_IPAR_WRITE_MPS_OBJ_SENSE` 242
  Controls the output file data.

- `MSK_IPAR_WRITE_MPS_QUOTED_NAMES` 242
  Controls the output file data.

- `MSK_IPAR_WRITE_MPS_STRICT` 243
  Controls the output MPS file format.

- `MSK_IPAR_WRITE_PRECISION` 243
  Controls data precision employed in when writing an MPS file.

- `MSK_IPAR_WRITE_SOL_CONSTRAINTS` 243
  Controls the solution file format.

- `MSK_IPAR_WRITE_SOL_HEAD` 244
  Controls solution file format.

- `MSK_IPAR_WRITE_SOL_VARIABLES` 244
  Controls the solution file format.

- `MSK_IPAR_WRITE_TASK_INC_SOL` 244
  Controls whether the solutions are stored in the task file too.

- `MSK_IPAR_WRITE_XML_MODE` 244
  Controls if linear coefficients should be written by row or column when writing in the XML file format.

- `alloc_add_qnz`

  Corresponding constant:
  `MSK_IPAR_ALLOC_ADD_QNZ`

  Description:
  Additional number of Q non-zeros that are allocated space for when `numanz` exceeds `maxnumqnz` during addition of new Q entries.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  5000

- `bi_clean_optimizer`

  Corresponding constant:
  `MSK_IPAR_BI_CLEAN_OPTIMIZER`

  Description:
  Controls which simplex optimizer is used in the clean-up phase.

Possible Values:

`MSK_OPTIMIZER_INTPNT` The interior-point optimizer is used.

`MSK_OPTIMIZER_CONCURRENT` The optimizer for nonconvex nonlinear problems.

`MSK_OPTIMIZER_MIXED_INT` The mixed integer optimizer.

`MSK_OPTIMIZER_DUAL_SIMPLEX` The dual simplex optimizer is used.

`MSK_OPTIMIZER_FREE` The optimizer is chosen automatically.

`MSK_OPTIMIZER_CONIC` Another cone optimizer.

`MSK_OPTIMIZER_NONCONVEX` The optimizer for nonconvex nonlinear problems.

`MSK_OPTIMIZER_QCONE` The Qcone optimizer is used.

`MSK_OPTIMIZER_PRIMAL_SIMPLEX` The primal simplex optimizer is used.

`MSK_OPTIMIZER_FREE_SIMPLEX` Either the primal or the dual simplex optimizer is used.

Default value:

MSK_OPTIMIZER_FREE

- `bi_ignore_max_iter`

Corresponding constant:

`MSK_IPAR_BI_IGNORE_MAX_ITER`

Description:

If the parameter `MSK_IPAR_INTPNT_BASIS` has the value `MSK_BI_NO_ERROR` and the interior-point optimizer has terminated due to maximum number of iterations, then basis identification is performed if this parameter has the value `MSK_ON`.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

MSK_OFF

- `bi_ignore_num_error`

Corresponding constant:

`MSK_IPAR_BI_IGNORE_NUM_ERROR`

Description:

If the parameter `MSK_IPAR_INTPNT_BASIS` has the value `MSK_BI_NO_ERROR` and the interior-point optimizer has terminated due to a numerical problem, then basis identification is performed if this parameter has the value `MSK_ON`.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

MSK_OFF

- `bi_max_iterations`

Corresponding constant:
`MSK_IPAR_BI_MAX_ITERATIONS`

Description:
Controls the maximum number of simplex iterations allowed to optimize a basis after the basis identification.

Possible Values:
Any number between 0 and +inf.

Default value:
1000000

- `cache_size_l1`

  Corresponding constant:
  `MSK_IPAR_CACHE_SIZE_L1`

  Description:
  Specifies the size of the cache of the computer. This parameter is potentially very important for the efficiency on computers if MOSEK cannot determine the cache size automatically. If the cache size is negative, then MOSEK tries to determine the value automatically.

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1

- `cache_size_l2`

  Corresponding constant:
  `MSK_IPAR_CACHE_SIZE_L2`

  Description:
  Specifies the size of the cache of the computer. This parameter is potentially very important for the efficiency on computers where MOSEK cannot determine the cache size automatically. If the cache size is negative, then MOSEK tries to determine the value automatically.

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1

- `check_convexity`

  Corresponding constant:
  `MSK_IPAR_CHECK_CONVEXITY`

  Description:
  Specify the level of convexity check on quadratic problems

  Possible Values:

  `MSK_CHECK_CONVEXITY_SIMPLE` Perform simple and fast convexity check.

`MSK_CHECK_CONVEXITY_NONE` No convexity check.

Default value:
MSK_CHECK_CONVEXITY_SIMPLE

- `check_ctrl_c`

Corresponding constant:
`MSK_IPAR_CHECK_CTRL_C`

Description:
Specifies whether MOSEK should check for `<ctrl>+<c>` key presses. In case it has, then control is returned to the user program.

In case a user-defined ctrl-c function is defined then that is used to check for ctrl-c. Otherwise the system procedure `signal` is used.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `check_task_data`

Corresponding constant:
`MSK_IPAR_CHECK_TASK_DATA`

Description:
If this feature is turned on, then the task data is checked for bad values i.e. NaNs. before an optimization is performed.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `concurrent_num_optimizers`

Corresponding constant:
`MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS`

Description:
The maximum number of simultaneous optimizations that will be started by the concurrent optimizer.

Possible Values:
Any number between 0 and +inf.

Default value:
2

- `concurrent_priority_dual_simplex`

  Corresponding constant:
  `MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX`

  Description:
  Priority of the dual simplex algorithm when selecting solvers for concurrent optimization.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  2

- `concurrent_priority_free_simplex`

  Corresponding constant:
  `MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX`

  Description:
  Priority of the free simplex optimizer when selecting solvers for concurrent optimization.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  3

- `concurrent_priority_intpnt`

  Corresponding constant:
  `MSK_IPAR_CONCURRENT_PRIORITY_INTPNT`

  Description:
  Priority of the interior-point algorithm when selecting solvers for concurrent optimization.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  4

- `concurrent_priority_primal_simplex`

  Corresponding constant:
  `MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX`

  Description:
  Priority of the primal simplex algorithm when selecting solvers for concurrent optimization.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  1

- `cpu_type`

Corresponding constant:
`MSK_IPAR_CPU_TYPE`

Description:
Specifies the CPU type. By default MOSEK tries to auto detect the CPU type. Therefore, we recommend to change this parameter only if the auto detection does not work properly.

Possible Values:

`MSK_CPU_POWERPC_G5`	A G5 PowerPC CPU.
`MSK_CPU_INTEL_PM`	An Intel PM cpu.
`MSK_CPU_GENERIC`	An generic CPU type for the platform
`MSK_CPU_UNKNOWN`	An unknown CPU.
`MSK_CPU_AMD_OPTERON`	An AMD Opteron (64 bit).
`MSK_CPU_INTEL_ITANIUM2`	An Intel Itanium2.
`MSK_CPU_AMD_ATHLON`	An AMD Athlon.
`MSK_CPU_HP_PARISC20`	An HP PA RISC version 2.0 CPU.
`MSK_CPU_INTEL_P4`	An Intel Pentium P4 or Intel Xeon.
`MSK_CPU_INTEL_P3`	An Intel Pentium P3.
`MSK_CPU_INTEL_CORE2`	An Intel CORE2 cpu.

Default value:
MSK_CPU_UNKNOWN

- `data_check`

Corresponding constant:
`MSK_IPAR_DATA_CHECK`

Description:
If this option is turned on, then extensive data checking is enabled. It will slow down MOSEK but on the other hand help locating bugs.

Possible Values:

`MSK_ON`	Switch the option on.
`MSK_OFF`	Switch the option off.

Default value:
MSK_ON

- `feasrepair_optimize`

Corresponding constant:
`MSK_IPAR_FEASREPAIR_OPTIMIZE`

Description:
Controls which type of feasibility analysis is to be performed.

Possible Values:

`MSK_FEASREPAIR_OPTIMIZE_NONE`	Do not optimize the feasibility repair problem.

`MSK_FEASREPAIR_OPTIMIZE_COMBINED` Minimize with original objective subject to minimal weighted violation of bounds.

`MSK_FEASREPAIR_OPTIMIZE_PENALTY` Minimize weighted sum of violations.

Default value:
MSK_FEASREPAIR_OPTIMIZE_NONE

- `flush_stream_freq`

  Corresponding constant:
  `MSK_IPAR_FLUSH_STREAM_FREQ`

  Description:
  Controls how frequent the message and log streams are flushed. A value of 0 means that it is never flushed. Otherwise a larger value results in less frequent flushes.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  24

- `infeas_generic_names`

  Corresponding constant:
  `MSK_IPAR_INFEAS_GENERIC_NAMES`

  Description:
  Controls whether generic names are used when an infeasible subproblem is created.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `infeas_prefer_primal`

  Corresponding constant:
  `MSK_IPAR_INFEAS_PREFER_PRIMAL`

  Description:
  If both certificates of primal and dual infeasibility are supplied then only the primal is used when this option is turned on.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `infeas_report_auto`

Corresponding constant:
`MSK_IPAR_INFEAS_REPORT_AUTO`

Description:
Controls whether an infeasibility report is automatically produced after the optimization if the problem is primal or dual infeasible.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `infeas_report_level`

Corresponding constant:
`MSK_IPAR_INFEAS_REPORT_LEVEL`

Description:
Controls the amount of information presented in an infeasibility report. Higher values imply more information.

Possible Values:
Any number between 0 and +inf.

Default value:
1

- `intpnt_basis`

Corresponding constant:
`MSK_IPAR_INTPNT_BASIS`

Description:
Controls whether the interior-point optimizer also computes an optimal basis.

Possible Values:

`MSK_BI_ALWAYS` Basis identification is always performed even if the interior-point optimizer terminates abnormally.

`MSK_BI_NO_ERROR` Basis identification is performed if the interior-point optimizer terminates without an error.

`MSK_BI_NEVER` Never do basis identification.

`MSK_BI_IF_FEASIBLE` Basis identification is not performed if the interior-point optimizer terminates with a problem status saying that the problem is primal or dual infeasible.

`MSK_BI_OTHER` Try another BI method.

Default value:
MSK_BI_ALWAYS

See also:

`MSK_IPAR_BI_IGNORE_MAX_ITER` Turns on basis identification in case the interior-point optimizer is terminated due to maximum number of iterations.

`MSK_IPAR_BI_IGNORE_NUM_ERROR` Turns on basis identification in case the interior-point optimizer is terminated due to a numerical problem.

- `intpnt_diff_step`

  Corresponding constant:
  `MSK_IPAR_INTPNT_DIFF_STEP`

  Description:
  Controls whether different step sizes are allowed in the primal and dual space.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `intpnt_factor_debug_lvl`

  Corresponding constant:
  `MSK_IPAR_INTPNT_FACTOR_DEBUG_LVL`

  Description:
  Controls factorization debug level.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `intpnt_factor_method`

  Corresponding constant:
  `MSK_IPAR_INTPNT_FACTOR_METHOD`

  Description:
  Controls the method used to factor the Newton equation system.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `intpnt_max_iterations`

  Corresponding constant:
  `MSK_IPAR_INTPNT_MAX_ITERATIONS`

  Description:
  Controls the maximum number of iterations allowed in the interior-point optimizer.

  Possible Values:
  Any number between 0 and +inf.

Default value:
400

- `intpnt_max_num_cor`

  Corresponding constant:
  `MSK_IPAR_INTPNT_MAX_NUM_COR`

  Description:
  Controls the maximum number of correctors allowed by the multiple corrector procedure. A negative value means that MOSEK is making the choice.

  Possible Values:
  Any number between -1 and +inf.

  Default value:
  -1

- `intpnt_max_num_refinement_steps`

  Corresponding constant:
  `MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS`

  Description:
  Maximum number of steps to be used by the iterative refinement of the search direction. A negative value implies that the optimizer Chooses the maximum number of iterative refinement steps.

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1

- `intpnt_num_threads`

  Corresponding constant:
  `MSK_IPAR_INTPNT_NUM_THREADS`

  Description:
  Controls the number of threads employed by the interior-point optimizer.

  Possible Values:
  Any integer greater than 1.

  Default value:
  1

- `intpnt_off_col_trh`

  Corresponding constant:
  `MSK_IPAR_INTPNT_OFF_COL_TRH`

  Description:
  Controls how many offending columns are detected in the Jacobian of the constraint matrix.

  1 means aggressive detection, higher values mean less aggressive detection.

  0 means no detection.

Possible Values:
Any number between 0 and +inf.

Default value:
40

- `intpnt_order_method`

  Corresponding constant:
  `MSK_IPAR_INTPNT_ORDER_METHOD`

  Description:
  Controls the ordering strategy used by the interior-point optimizer when factorizing the Newton equation system.

  Possible Values:

  `MSK_ORDER_METHOD_NONE` No ordering is used.

  `MSK_ORDER_METHOD_APPMINLOC2` A variant of the approximate minimum local-fill-in ordering is used.

  `MSK_ORDER_METHOD_APPMINLOC1` Approximate minimum local-fill-in ordering is used.

  `MSK_ORDER_METHOD_GRAPHPAR2` An alternative graph partitioning based ordering.

  `MSK_ORDER_METHOD_FREE` The ordering method is chosen automatically.

  `MSK_ORDER_METHOD_GRAPHPAR1` Graph partitioning based ordering.

  Default value:
  MSK_ORDER_METHOD_FREE

- `intpnt_regularization_use`

  Corresponding constant:
  `MSK_IPAR_INTPNT_REGULARIZATION_USE`

  Description:
  Controls whether regularization is allowed.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `intpnt_scaling`

  Corresponding constant:
  `MSK_IPAR_INTPNT_SCALING`

  Description:
  Controls how the problem is scaled before the interior-point optimizer is used.

  Possible Values:

  `MSK_SCALING_NONE` No scaling is performed.

`MSK_SCALING_MODERATE` A conservative scaling is performed.

`MSK_SCALING_AGGRESSIVE` A very aggressive scaling is performed.

`MSK_SCALING_FREE` The optimizer chooses the scaling heuristic.

Default value:
MSK_SCALING_FREE

- `intpnt_solve_form`

Corresponding constant:
`MSK_IPAR_INTPNT_SOLVE_FORM`

Description:
Controls whether the primal or the dual problem is solved.

Possible Values:

`MSK_SOLVE_PRIMAL` The optimizer should solve the primal problem.

`MSK_SOLVE_DUAL` The optimizer should solve the dual problem.

`MSK_SOLVE_FREE` The optimizer is free to solve either the primal or the dual problem.

Default value:
MSK_SOLVE_FREE

- `intpnt_starting_point`

Corresponding constant:
`MSK_IPAR_INTPNT_STARTING_POINT`

Description:
Starting point used by the interior-point optimizer.

Possible Values:

`MSK_STARTING_POINT_CONSTANT` The starting point is set to a constant. This is more reliable than a non-constant starting point.

`MSK_STARTING_POINT_FREE` The starting point is chosen automatically.

Default value:
MSK_STARTING_POINT_FREE

- `license_allow_overuse`

Corresponding constant:
`MSK_IPAR_LICENSE_ALLOW_OVERUSE`

Description:
Controls if license overuse is allowed when caching licenses

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `license_cache_time`

  Corresponding constant:
  `MSK_IPAR_LICENSE_CACHE_TIME`

  Description:
  Controls the amount of time a license is cached in the MOSEK environment for reuse. Checking out a license from the license server has a small overhead. Therefore, if a large number of optimizations is performed within a small amount of time, it is efficient to cache the license in the MOSEK environment for later use. This way a number of license check outs from the license server is avoided.

  If a license has not been used in the given amount of time,MOSEK will automatically check in the license. To disable license caching set the value to 0.

  Possible Values:
  Any number between 0 and 65555.

  Default value:
  5

- `license_check_time`

  Corresponding constant:
  `MSK_IPAR_LICENSE_CHECK_TIME`

  Description:
  The parameter specifies the number of seconds between the checks of all the active licenses in the MOSEK environment license cache. These checks are performed to determine if the licenses should be returned to the server.

  Possible Values:
  Any number between 1 and 120.

  Default value:
  1

- `license_debug`

  Corresponding constant:
  `MSK_IPAR_LICENSE_DEBUG`

  Description:
  This option is used to turn on debugging of the incense manager.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `license_pause_time`

  Corresponding constant:
  `MSK_IPAR_LICENSE_PAUSE_TIME`

Description:
If `MSK_IPAR_LICENSE_WAIT`=`MSK_ON` and no license is available, then MOSEK sleeps a number of micro seconds between each check of whether a license as become free.

Possible Values:
Any number between 0 and 1000000.

Default value:
100

- `license_suppress_expire_wrns`

  Corresponding constant:
  `MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS`

  Description:
  Controls whether license features expire warnings are suppressed.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `license_wait`

  Corresponding constant:
  `MSK_IPAR_LICENSE_WAIT`

  Description:
  If all licenses are in use MOSEK returns with an error code. However, by turning on this parameter MOSEK will wait for an available license.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `log`

  Corresponding constant:
  `MSK_IPAR_LOG`

  Description:
  Controls the amount of log information. The value 0 implies that all log information is suppressed. A higher level implies that more information is logged.

  Please note that if a task is employed to solve a sequence of optimization problems the value of this parameter is reduced by the value of `MSK_IPAR_LOG_CUT_SECOND_OPT` for the second and any subsequent optimizations.

Possible Values:
Any number between 0 and +inf.

Default value:
10

See also:

`MSK_IPAR_LOG_CUT_SECOND_OPT` Controls the reduction in the log levels for the second and any subsequent optimizations.

- `log_bi`

  Corresponding constant:
  `MSK_IPAR_LOG_BI`

  Description:
  Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  4

- `log_bi_freq`

  Corresponding constant:
  `MSK_IPAR_LOG_BI_FREQ`

  Description:
  Controls how frequent the optimizer outputs information about the basis identification and how frequent the user-defined call-back function is called.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  2500

- `log_concurrent`

  Corresponding constant:
  `MSK_IPAR_LOG_CONCURRENT`

  Description:
  Controls amount of output printed by the concurrent optimizer.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  1

- `log_cut_second_opt`

Corresponding constant:
`MSK_IPAR_LOG_CUT_SECOND_OPT`

Description:
If a task is employed to solve a sequence of optimization problems, then the value of the log levels is reduced by the value of this parameter. E.g `MSK_IPAR_LOG` and `MSK_IPAR_LOG_SIM` are reduced by the value of this parameter for the second and any subsequent optimizations.

Possible Values:
Any number between 0 and +inf.

Default value:
1

See also:

`MSK_IPAR_LOG` Controls the amount of log information.

`MSK_IPAR_LOG_INTPNT` Controls the amount of log information from the interior-point optimizers.

`MSK_IPAR_LOG_MIO` Controls the amount of log information from the mixed-integer optimizers.

`MSK_IPAR_LOG_SIM` Controls the amount of log information from the simplex optimizers.

- `log_factor`

Corresponding constant:
`MSK_IPAR_LOG_FACTOR`

Description:
If turned on, then the factor log lines are added to the log.

Possible Values:
Any number between 0 and +inf.

Default value:
1

- `log_feasrepair`

Corresponding constant:
`MSK_IPAR_LOG_FEASREPAIR`

Description:
Controls the amount of output printed when performing feasibility repair.

Possible Values:
Any number between 0 and +inf.

Default value:
0

- `log_file`

Corresponding constant:
`MSK_IPAR_LOG_FILE`

Description:
If turned on, then some log info is printed when a file is written or read.

Possible Values:
Any number between 0 and +inf.

Default value:
1

- `log_head`

  Corresponding constant:
  `MSK_IPAR_LOG_HEAD`

  Description:
  If turned on, then a header line is added to the log.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  1

- `log_infeas_ana`

  Corresponding constant:
  `MSK_IPAR_LOG_INFEAS_ANA`

  Description:
  Controls amount of output printed by the infeasibility analyzer procedures. A higher level implies that more information is logged.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  1

- `log_intpnt`

  Corresponding constant:
  `MSK_IPAR_LOG_INTPNT`

  Description:
  Controls amount of output printed printed by the interior-point optimizer. A higher level implies that more information is logged.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  4

- `log_mio`

  Corresponding constant:
  `MSK_IPAR_LOG_MIO`

Description:
Controls the log level for the mixed integer optimizer. A higher level implies that more information is logged.

Possible Values:
Any number between 0 and +inf.

Default value:
4

- `log_mio_freq`

  Corresponding constant:
  `MSK_IPAR_LOG_MIO_FREQ`

  Description:
  Controls how frequent the mixed integer optimizer prints the log line. It will print line every time `MSK_IPAR_LOG_MIO_FREQ` relaxations have been solved.

  Possible Values:
  A integer value.

  Default value:
  250

- `log_nonconvex`

  Corresponding constant:
  `MSK_IPAR_LOG_NONCONVEX`

  Description:
  Controls amount of output printed by the nonconvex optimizer.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  1

- `log_optimizer`

  Corresponding constant:
  `MSK_IPAR_LOG_OPTIMIZER`

  Description:
  Controls the amount of general optimizer information that is logged.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  1

- `log_order`

  Corresponding constant:
  `MSK_IPAR_LOG_ORDER`

Description:
If turned on, then factor lines are added to the log.

Possible Values:
Any number between 0 and +inf.

Default value:
1

- `log_param`

  Corresponding constant:
  `MSK_IPAR_LOG_PARAM`

  Description:
  Controls the amount of information printed out about parameter changes.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `log_presolve`

  Corresponding constant:
  `MSK_IPAR_LOG_PRESOLVE`

  Description:
  Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  1

- `log_response`

  Corresponding constant:
  `MSK_IPAR_LOG_RESPONSE`

  Description:
  Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `log_sensitivity`

  Corresponding constant:
  `MSK_IPAR_LOG_SENSITIVITY`

Description:
Controls the amount of logging during the sensitivity analysis. 0: Means no logging information is produced. 1: Timing information is printed. 2: Sensitivity results are printed.

Possible Values:
Any number between 0 and +inf.

Default value:
1

- `log_sensitivity_opt`

  Corresponding constant:
  `MSK_IPAR_LOG_SENSITIVITY_OPT`

  Description:
  Controls the amount of logging from the optimizers employed during the sensitivity analysis. 0 means no logging information is produced.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `log_sim`

  Corresponding constant:
  `MSK_IPAR_LOG_SIM`

  Description:
  Controls amount of output printed by the simplex optimizer. A higher level implies that more information is logged.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  4

- `log_sim_freq`

  Corresponding constant:
  `MSK_IPAR_LOG_SIM_FREQ`

  Description:
  Controls how frequent the simplex optimizer outputs information about the optimization and how frequent the user-defined call-back function is called.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  500

- `log_sim_minor`

Corresponding constant:
`MSK_IPAR_LOG_SIM_MINOR`

Description:
Currently not in use.

Possible Values:
Any number between 0 and +inf.

Default value:
1

- `log_sim_network_freq`

  Corresponding constant:
  `MSK_IPAR_LOG_SIM_NETWORK_FREQ`

  Description:
  Controls how frequent the network simplex optimizer outputs information about the optimization and how frequent the user-defined call-back function is called. The network optimizer will use a logging frequency equal to `MSK_IPAR_LOG_SIM_FREQ` times `MSK_IPAR_LOG_SIM_NETWORK_FREQ`.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  50

- `log_storage`

  Corresponding constant:
  `MSK_IPAR_LOG_STORAGE`

  Description:
  When turned on, MOSEK prints messages regarding the storage usage and allocation.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `lp_write_ignore_incompatible_items`

  Corresponding constant:
  `MSK_IPAR_LP_WRITE_IGNORE_INCOMPATIBLE_ITEMS`

  Description:
  Controls the result of writing a problem containing incompatible items to an LP file.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `max_num_warnings`

  Corresponding constant:
  `MSK_IPAR_MAX_NUM_WARNINGS`

  Description:
  Waning level. A higher value results in more warnings.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  10

- `maxnumanz_double_trh`

  Corresponding constant:
  `MSK_IPAR_MAXNUMANZ_DOUBLE_TRH`

  Description:
  Whenever MOSEK runs out of storage for the A matrix, it will double the value for `maxnumanz` until `maxnumnza` reaches the value of this parameter. When this threshold is reached it will use a slower increase.

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1

- `mio_branch_dir`

  Corresponding constant:
  `MSK_IPAR_MIO_BRANCH_DIR`

  Description:
  Controls whether the mixed integer optimizer is branching up or down by default.

  Possible Values:

  `MSK_BRANCH_DIR_DOWN` The mixed integer optimizer always chooses the down branch first.

  `MSK_BRANCH_DIR_UP` The mixed integer optimizer always chooses the up branch first.

  `MSK_BRANCH_DIR_FREE` The mixed optimizer decides which branch to choose.

  Default value:
  MSK_BRANCH_DIR_FREE

- `mio_branch_priorities_use`

  Corresponding constant:
  `MSK_IPAR_MIO_BRANCH_PRIORITIES_USE`

  Description:
  Controls whether branching priorities are used by the mixed integer optimizer.

  Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

MSK_ON

- `mio_construct_sol`

Corresponding constant:

`MSK_IPAR_MIO_CONSTRUCT_SOL`

Description:

If set to `MSK_ON` and all integer variables have been given a value for which a feasible MIP solution exists, then MOSEK generates an initial solution to the MIP by fixing all integer values and solving for the continuous variables.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

MSK_OFF

- `mio_cont_sol`

Corresponding constant:

`MSK_IPAR_MIO_CONT_SOL`

Description:

Controls the meaning of the interior-point and basic solutions in MIP problems.

Possible Values:

`MSK_MIO_CONT_SOL_ITG` The reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. A solution is only reported in case the problem has a primal feasible solution.

`MSK_MIO_CONT_SOL_NONE` No interior-point or basic solution are reported when the mixed integer optimizer is used.

`MSK_MIO_CONT_SOL_ROOT` The reported interior-point and basic solutions are a solution to the root node problem when mixed integer optimizer is used.

`MSK_MIO_CONT_SOL_ITG_REL` In case the problem is primal feasible then the reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. If the problem is primal infeasible, then the solution to the root node problem is reported.

Default value:

MSK_MIO_CONT_SOL_NONE

- `mio_cut_level_root`

Corresponding constant:

`MSK_IPAR_MIO_CUT_LEVEL_ROOT`

Description:

Controls the cut level employed by the mixed integer optimizer at the root node. A negative value means a default value determined by the mixed integer optimizer is used. By adding the appropriate values from the following table the employed cut types can be controlled.

GUB cover	+2
Flow cover	+4
Lifting	+8
Plant location	+16
Disaggregation	+32
Knapsack cover	+64
Lattice	+128
Gomory	+256
Coefficient reduction	+512
GCD	+1024
Obj. integrality	+2048

Possible Values:

Any value.

Default value:

-1

- `mio_cut_level_tree`

  Corresponding constant:

  `MSK_IPAR_MIO_CUT_LEVEL_TREE`

  Description:

  Controls the cut level employed by the mixed integer optimizer at the tree. See `MSK_IPAR_MIO_CUT_LEVEL_ROOT` for an explanation of the parameter values.

  Possible Values:

  Any value.

  Default value:

  -1

- `mio_feaspump_level`

  Corresponding constant:

  `MSK_IPAR_MIO_FEASPUMP_LEVEL`

  Description:

  Feasibility pump is a heuristic designed to compute an initial feasible solution. A value of 0 implies that the feasibility pump heuristic is not used. A value of -1 implies that the mixed integer optimizer decides how the feasibility pump heuristic is used. A larger value than 1 implies that the feasibility pump is employed more aggressively. Normally a value beyond 3 is not worthwhile.

  Possible Values:

  Any number between -inf and 3.

  Default value:

  -1

- `mio_heuristic_level`

  Corresponding constant:
  `MSK_IPAR_MIO_HEURISTIC_LEVEL`

  Description:
  Controls the heuristic employed by the mixed integer optimizer to locate an initial good integer feasible solution. A value of zero means the heuristic is not used at all. A larger value than 0 means that a gradually more sophisticated heuristic is used which is computationally more expensive. A negative value implies that the optimizer chooses the heuristic. Normally a value around 3 to 5 should be optimal.

  Possible Values:
  Any value.

  Default value:
  -1

- `mio_keep_basis`

  Corresponding constant:
  `MSK_IPAR_MIO_KEEP_BASIS`

  Description:
  Controls whether the integer presolve keeps bases in memory. This speeds on the solution process at cost of bigger memory consumption.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `mio_local_branch_number`

  Corresponding constant:
  `MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER`

  Description:

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1

- `mio_max_num_branches`

  Corresponding constant:
  `MSK_IPAR_MIO_MAX_NUM_BRANCHES`

  Description:
  Maximum number of branches allowed during the branch and bound search. A negative value means infinite.

Possible Values:
Any number between -inf and +inf.

Default value:
-1

See also:

`MSK_DPAR_MIO_DISABLE_TERM_TIME` Certain termination criterias is disabled within the mixed integer optimizer for period time specified by the parameter.

- `mio_max_num_relaxs`

Corresponding constant:
`MSK_IPAR_MIO_MAX_NUM_RELAXS`

Description:
Maximum number of relaxations allowed during the branch and bound search. A negative value means infinite.

Possible Values:
Any number between -inf and +inf.

Default value:
-1

See also:

`MSK_DPAR_MIO_DISABLE_TERM_TIME` Certain termination criterias is disabled within the mixed integer optimizer for period time specified by the parameter.

- `mio_max_num_solutions`

Corresponding constant:
`MSK_IPAR_MIO_MAX_NUM_SOLUTIONS`

Description:
The mixed integer optimizer can be terminated after a certain number of different feasible solutions has been located. If this parameter has the value n and n is strictly positive, then the mixed integer optimizer will be terminated when n feasible solutions have been located.

Possible Values:
Any number between -inf and +inf.

Default value:
-1

See also:

`MSK_DPAR_MIO_DISABLE_TERM_TIME` Certain termination criterias is disabled within the mixed integer optimizer for period time specified by the parameter.

- `mio_mode`

Corresponding constant:
`MSK_IPAR_MIO_MODE`

Description:
Controls whether the optimizer includes the integer restrictions when solving a (mixed) integer optimization problem.

Possible Values:

`MSK_MIO_MODE_IGNORED` The integer constraints are ignored and the problem is solved as a continuous problem.

`MSK_MIO_MODE_LAZY` Integer restrictions should be satisfied if an optimizer is available for the problem.

`MSK_MIO_MODE_SATISFIED` Integer restrictions should be satisfied.

Default value:
MSK_MIO_MODE_SATISFIED

- `mio_node_optimizer`

Corresponding constant:
`MSK_IPAR_MIO_NODE_OPTIMIZER`

Description:
Controls which optimizer is employed at the non-root nodes in the mixed integer optimizer.

Possible Values:

`MSK_OPTIMIZER_INTPNT` The interior-point optimizer is used.

`MSK_OPTIMIZER_CONCURRENT` The optimizer for nonconvex nonlinear problems.

`MSK_OPTIMIZER_MIXED_INT` The mixed integer optimizer.

`MSK_OPTIMIZER_DUAL_SIMPLEX` The dual simplex optimizer is used.

`MSK_OPTIMIZER_FREE` The optimizer is chosen automatically.

`MSK_OPTIMIZER_CONIC` Another cone optimizer.

`MSK_OPTIMIZER_NONCONVEX` The optimizer for nonconvex nonlinear problems.

`MSK_OPTIMIZER_QCONE` The Qcone optimizer is used.

`MSK_OPTIMIZER_PRIMAL_SIMPLEX` The primal simplex optimizer is used.

`MSK_OPTIMIZER_FREE_SIMPLEX` Either the primal or the dual simplex optimizer is used.

Default value:
MSK_OPTIMIZER_FREE

- `mio_node_selection`

Corresponding constant:
`MSK_IPAR_MIO_NODE_SELECTION`

Description:
Controls the node selection strategy employed by the mixed integer optimizer.

Possible Values:

`MSK_MIO_NODE_SELECTION_PSEUDO` The optimizer employs selects the node based on a pseudo cost estimate.

`MSK_MIO_NODE_SELECTION_HYBRID` The optimizer employs a hybrid strategy.

`MSK_MIO_NODE_SELECTION_FREE` The optimizer decides the node selection strategy.

`MSK_MIO_NODE_SELECTION_WORST` The optimizer employs a worst bound node selection strategy.

`MSK_MIO_NODE_SELECTION_BEST` The optimizer employs a best bound node selection strategy.

`MSK_MIO_NODE_SELECTION_FIRST` The optimizer employs a depth first node selection strategy.

Default value:

MSK_MIO_NODE_SELECTION_FREE

- `mio_presolve_aggregate`

  Corresponding constant:

  `MSK_IPAR_MIO_PRESOLVE_AGGREGATE`

  Description:

  Controls whether the presolve used by the mixed integer optimizer tries to aggregate the constraints.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:

  MSK_ON

- `mio_presolve_probing`

  Corresponding constant:

  `MSK_IPAR_MIO_PRESOLVE_PROBING`

  Description:

  Controls whether the mixed integer presolve performs probing. Probing can be very time consuming.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:

  MSK_ON

- `mio_presolve_use`

  Corresponding constant:

  `MSK_IPAR_MIO_PRESOLVE_USE`

  Description:

  Controls whether presolve is performed by the mixed integer optimizer.

  Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `mio_root_optimizer`

  Corresponding constant:
  `MSK_IPAR_MIO_ROOT_OPTIMIZER`

  Description:
  Controls which optimizer is employed at the root node in the mixed integer optimizer.

  Possible Values:

  `MSK_OPTIMIZER_INTPNT` The interior-point optimizer is used.

  `MSK_OPTIMIZER_CONCURRENT` The optimizer for nonconvex nonlinear problems.

  `MSK_OPTIMIZER_MIXED_INT` The mixed integer optimizer.

  `MSK_OPTIMIZER_DUAL_SIMPLEX` The dual simplex optimizer is used.

  `MSK_OPTIMIZER_FREE` The optimizer is chosen automatically.

  `MSK_OPTIMIZER_CONIC` Another cone optimizer.

  `MSK_OPTIMIZER_NONCONVEX` The optimizer for nonconvex nonlinear problems.

  `MSK_OPTIMIZER_QCONE` The Qcone optimizer is used.

  `MSK_OPTIMIZER_PRIMAL_SIMPLEX` The primal simplex optimizer is used.

  `MSK_OPTIMIZER_FREE_SIMPLEX` Either the primal or the dual simplex optimizer is used.

  Default value:
  MSK_OPTIMIZER_FREE

- `mio_strong_branch`

  Corresponding constant:
  `MSK_IPAR_MIO_STRONG_BRANCH`

  Description:
  The value specifies the depth from the root in which strong branching is used. A negative value means that the optimizer chooses a default value automatically.

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1

- `nonconvex_max_iterations`

  Corresponding constant:
  `MSK_IPAR_NONCONVEX_MAX_ITERATIONS`

  Description:
  Maximum number of iterations that can be used by the nonconvex optimizer.

Possible Values:
Any number between 0 and +inf.

Default value:
100000

- `objective_sense`

  Corresponding constant:
  `MSK_IPAR_OBJECTIVE_SENSE`

  Description:
  If the objective sense for the task is undefined, then the value of this parameter is used as the default objective sense.

  Possible Values:

  `MSK_OBJECTIVE_SENSE_MINIMIZE` The problem should be minimized.

  `MSK_OBJECTIVE_SENSE_UNDEFINED` The objective sense is undefined.

  `MSK_OBJECTIVE_SENSE_MAXIMIZE` The problem should be maximized.

  Default value:
  MSK_OBJECTIVE_SENSE_MINIMIZE

- `opf_max_terms_per_line`

  Corresponding constant:
  `MSK_IPAR_OPF_MAX_TERMS_PER_LINE`

  Description:
  The maximum number of terms (linear and quadratic) per line when an OPF file is written.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  5

- `opf_write_header`

  Corresponding constant:
  `MSK_IPAR_OPF_WRITE_HEADER`

  Description:
  Write a text header with date and MOSEK version in an OPF file.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `opf_write_hints`

Corresponding constant:
`MSK_IPAR_OPF_WRITE_HINTS`

Description:
Write a hint section with problem dimensions in the beginning of an OPF file.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `opf_write_parameters`

Corresponding constant:
`MSK_IPAR_OPF_WRITE_PARAMETERS`

Description:
Write a parameter section in an OPF file.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `opf_write_problem`

Corresponding constant:
`MSK_IPAR_OPF_WRITE_PROBLEM`

Description:
Write objective, constraints, bounds etc. to an OPF file.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `opf_write_sol_bas`

Corresponding constant:
`MSK_IPAR_OPF_WRITE_SOL_BAS`

Description:
If `MSK_IPAR_OPF_WRITE_SOLUTIONS` is `MSK_ON` and a basic solution is defined, include the basic solution in OPF files.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `opf_write_sol_itg`

  Corresponding constant:
  `MSK_IPAR_OPF_WRITE_SOL_ITG`

  Description:
  If `MSK_IPAR_OPF_WRITE_SOLUTIONS` is `MSK_ON` and an integer solution is defined, write the integer solution in OPF files.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `opf_write_sol_itr`

  Corresponding constant:
  `MSK_IPAR_OPF_WRITE_SOL_ITR`

  Description:
  If `MSK_IPAR_OPF_WRITE_SOLUTIONS` is `MSK_ON` and an interior solution is defined, write the interior solution in OPF files.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `opf_write_solutions`

  Corresponding constant:
  `MSK_IPAR_OPF_WRITE_SOLUTIONS`

  Description:
  Enable inclusion of solutions in the OPF files.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `optimizer`

Corresponding constant:
`MSK_IPAR_OPTIMIZER`

Description:
Controls which optimizer is used to optimize the task.

Possible Values:

`MSK_OPTIMIZER_INTPNT` The interior-point optimizer is used.

`MSK_OPTIMIZER_CONCURRENT` The optimizer for nonconvex nonlinear problems.

`MSK_OPTIMIZER_MIXED_INT` The mixed integer optimizer.

`MSK_OPTIMIZER_DUAL_SIMPLEX` The dual simplex optimizer is used.

`MSK_OPTIMIZER_FREE` The optimizer is chosen automatically.

`MSK_OPTIMIZER_CONIC` Another cone optimizer.

`MSK_OPTIMIZER_NONCONVEX` The optimizer for nonconvex nonlinear problems.

`MSK_OPTIMIZER_QCONE` The Qcone optimizer is used.

`MSK_OPTIMIZER_PRIMAL_SIMPLEX` The primal simplex optimizer is used.

`MSK_OPTIMIZER_FREE_SIMPLEX` Either the primal or the dual simplex optimizer is used.

Default value:
MSK_OPTIMIZER_FREE

- `param_read_case_name`

Corresponding constant:
`MSK_IPAR_PARAM_READ_CASE_NAME`

Description:
If turned on, then names in the parameter file are case sensitive.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `param_read_ign_error`

Corresponding constant:
`MSK_IPAR_PARAM_READ_IGN_ERROR`

Description:
If turned on, then errors in paramter settings is ignored.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `presolve_elim_fill`

  Corresponding constant:
  `MSK_IPAR_PRESOLVE_ELIM_FILL`

  Description:
  Controls the maximum amount of fill-in that can be created during the elimination phase of the presolve. This parameter times (`numcon`+`numvar`) denotes the amount of fill-in.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  1

- `presolve_eliminator_use`

  Corresponding constant:
  `MSK_IPAR_PRESOLVE_ELIMINATOR_USE`

  Description:
  Controls whether free or implied free variables are eliminated from the problem.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `presolve_level`

  Corresponding constant:
  `MSK_IPAR_PRESOLVE_LEVEL`

  Description:
  Currently not used.

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1

- `presolve_lindep_use`

  Corresponding constant:
  `MSK_IPAR_PRESOLVE_LINDEP_USE`

  Description:
  Controls whether the linear constraints are checked for linear dependencies.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `presolve_lindep_work_lim`

Corresponding constant:
`MSK_IPAR_PRESOLVE_LINDEP_WORK_LIM`

Description:
Is used to limit the amount of work that can done to locate linear dependencies. In general the higher value this parameter is given the less work can be used. However, a value of 0 means no limit on the amount work that can be used.

Possible Values:
Any number between 0 and +inf.

Default value:
1

- `presolve_use`

Corresponding constant:
`MSK_IPAR_PRESOLVE_USE`

Description:
Controls whether the presolve is applied to a problem before it is optimized.

Possible Values:

`MSK_PRESOLVE_MODE_ON` The problem is presolved before it is optimized.

`MSK_PRESOLVE_MODE_OFF` The problem is not presolved before it is optimized.

`MSK_PRESOLVE_MODE_FREE` It is decided automatically whether to presolve before the problem is optimized.

Default value:
MSK_PRESOLVE_MODE_FREE

- `read_add_anz`

Corresponding constant:
`MSK_IPAR_READ_ADD_ANZ`

Description:
Additional number of non-zeros in A that is made room for in the problem.

Possible Values:
Any number between 0 and +inf.

Default value:
0

- `read_add_con`

Corresponding constant:
`MSK_IPAR_READ_ADD_CON`

Description:
Additional number of constraints that is made room for in the problem.

Possible Values:
Any number between 0 and +inf.

Default value:
0

- `read_add_cone`

  Corresponding constant:
  `MSK_IPAR_READ_ADD_CONE`

  Description:
  Additional number of conic constraints that is made room for in the problem.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `read_add_qnz`

  Corresponding constant:
  `MSK_IPAR_READ_ADD_QNZ`

  Description:
  Additional number of non-zeros in the Q matrices that is made room for in the problem.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `read_add_var`

  Corresponding constant:
  `MSK_IPAR_READ_ADD_VAR`

  Description:
  Additional number of variables that is made room for in the problem.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `read_anz`

  Corresponding constant:
  `MSK_IPAR_READ_ANZ`

Description:
Expected maximum number of A non-zeros to be read. The option is used only by fast MPS and LP file readers.

Possible Values:
Any number between 0 and +inf.

Default value:
100000

- `read_con`

  Corresponding constant:
  `MSK_IPAR_READ_CON`

  Description:
  Expected maximum number of constraints to be read. The option is only used by fast MPS and LP file readers.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  10000

- `read_cone`

  Corresponding constant:
  `MSK_IPAR_READ_CONE`

  Description:
  Expected maximum number of conic constraints to be read. The option is used only by fast MPS and LP file readers.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  2500

- `read_data_compressed`

  Corresponding constant:
  `MSK_IPAR_READ_DATA_COMPRESSED`

  Description:
  If this option is turned on,it is assumed that the data file is compressed.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `read_data_format`

Corresponding constant:
`MSK_IPAR_READ_DATA_FORMAT`

Description:
Format of the data file to be read.

Possible Values:

`MSK_DATA_FORMAT_XML` The data file is an XML formatted file.

`MSK_DATA_FORMAT_EXTENSION` The file extension is used to determine the data file format.

`MSK_DATA_FORMAT_MPS` The data file is MPS formatted.

`MSK_DATA_FORMAT_LP` The data file is LP formatted.

`MSK_DATA_FORMAT_MBT` The data file is a MOSEK binary task file.

`MSK_DATA_FORMAT_OP` The data file is an optimization problem formatted file.

Default value:
MSK_DATA_FORMAT_EXTENSION

- `read_keep_free_con`

Corresponding constant:
`MSK_IPAR_READ_KEEP_FREE_CON`

Description:
Controls whether the free constraints are included in the problem.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `read_lp_drop_new_vars_in_bou`

Corresponding constant:
`MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU`

Description:
If this option is turned on, MOSEK will drop variables that are defined for the first time in the bounds section.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `read_lp_quoted_names`

Corresponding constant:
`MSK_IPAR_READ_LP_QUOTED_NAMES`

Description:
If a name is in quotes when reading an LP file, the quotes will be removed.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `read_mps_format`

Corresponding constant:
`MSK_IPAR_READ_MPS_FORMAT`

Description:
Controls how strictly the MPS file reader interprets the MPS format.

Possible Values:

`MSK_MPS_FORMAT_STRICT` It is assumed that the input file satisfies the MPS format strictly.

`MSK_MPS_FORMAT_RELAXED` It is assumed that the input file satisfies a slightly relaxed version of the MPS format.

`MSK_MPS_FORMAT_FREE` It is assumed that the input file satisfies the free MPS format. This implies that spaces are not allowed in names. Otherwise the format is free.

Default value:
MSK_MPS_FORMAT_RELAXED

- `read_mps_keep_int`

Corresponding constant:
`MSK_IPAR_READ_MPS_KEEP_INT`

Description:
Controls whether MOSEK should keep the integer restrictions on the variables while reading the MPS file.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `read_mps_obj_sense`

Corresponding constant:
`MSK_IPAR_READ_MPS_OBJ_SENSE`

Description:
If turned on, the MPS reader uses the objective sense section. Otherwise the MPS reader ignores it.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `read_mps_quoted_names`

  Corresponding constant:
  `MSK_IPAR_READ_MPS_QUOTED_NAMES`

  Description:
  If a name is in quotes when reading an MPS file, then the quotes will be removed.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `read_mps_relax`

  Corresponding constant:
  `MSK_IPAR_READ_MPS_RELAX`

  Description:
  If this option is turned on, then the relaxation of the MIP will be read.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `read_mps_width`

  Corresponding constant:
  `MSK_IPAR_READ_MPS_WIDTH`

  Description:
  Controls the maximal number of chars allowed in one line of the MPS file.

  Possible Values:
  Any positive number greater than 80.

  Default value:
  1024

- `read_q_mode`

  Corresponding constant:
  `MSK_IPAR_READ_Q_MODE`

Description:
Controls how the Q matrices are read from the MPS file.

Possible Values:

`MSK_Q_READ_ADD` All elements in a Q matrix are assumed to belong to the lower triangular part. Duplicate elements in a Q matrix are added together.

`MSK_Q_READ_DROP_LOWER` All elements in the strict lower triangular part of the Q matrices are dropped.

`MSK_Q_READ_DROP_UPPER` All elements in the strict upper triangular part of the Q matrices are dropped.

Default value:
MSK_Q_READ_ADD

- `read_qnz`

Corresponding constant:
`MSK_IPAR_READ_QNZ`

Description:
Expected maximum number of Q non-zeros to be read. The option is used only by MPS and LP file readers.

Possible Values:
Any number between 0 and +inf.

Default value:
20000

- `read_task_ignore_param`

Corresponding constant:
`MSK_IPAR_READ_TASK_IGNORE_PARAM`

Description:
Controls whether MOSEK should ignore the parameter setting defined in the task file and use the default parameter setting instead.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `read_var`

Corresponding constant:
`MSK_IPAR_READ_VAR`

Description:
Expected maximum number of variable to be read. The option is used only by MPS and LP file readers.

Possible Values:
Any number between 0 and +inf.

Default value:
10000

- `sensitivity_all`

  Corresponding constant:
  `MSK_IPAR_SENSITIVITY_ALL`

  Description:

  Not applicable.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `sensitivity_optimizer`

  Corresponding constant:
  `MSK_IPAR_SENSITIVITY_OPTIMIZER`

  Description:
  Controls which optimizer is used for optimal partition sensitivity analysis.

  Possible Values:

  `MSK_OPTIMIZER_INTPNT` The interior-point optimizer is used.

  `MSK_OPTIMIZER_CONCURRENT` The optimizer for nonconvex nonlinear problems.

  `MSK_OPTIMIZER_MIXED_INT` The mixed integer optimizer.

  `MSK_OPTIMIZER_DUAL_SIMPLEX` The dual simplex optimizer is used.

  `MSK_OPTIMIZER_FREE` The optimizer is chosen automatically.

  `MSK_OPTIMIZER_CONIC` Another cone optimizer.

  `MSK_OPTIMIZER_NONCONVEX` The optimizer for nonconvex nonlinear problems.

  `MSK_OPTIMIZER_QCONE` The Qcone optimizer is used.

  `MSK_OPTIMIZER_PRIMAL_SIMPLEX` The primal simplex optimizer is used.

  `MSK_OPTIMIZER_FREE_SIMPLEX` Either the primal or the dual simplex optimizer is used.

  Default value:
  MSK_OPTIMIZER_FREE_SIMPLEX

- `sensitivity_type`

  Corresponding constant:
  `MSK_IPAR_SENSITIVITY_TYPE`

  Description:
  Controls which type of sensitivity analysis is to be performed.

Possible Values:

`MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION` Optimal partition sensitivity analysis is performed.

`MSK_SENSITIVITY_TYPE_BASIS` Basis sensitivity analysis is performed.

Default value:
MSK_SENSITIVITY_TYPE_BASIS

- `sim_degen`

Corresponding constant:
`MSK_IPAR_SIM_DEGEN`

Description:
Controls how aggressive degeneration is approached.

Possible Values:

`MSK_SIM_DEGEN_NONE` The simplex optimizer should use no degeneration strategy.

`MSK_SIM_DEGEN_MODERATE` The simplex optimizer should use a moderate degeneration strategy.

`MSK_SIM_DEGEN_MINIMUM` The simplex optimizer should use a minimum degeneration strategy.

`MSK_SIM_DEGEN_AGGRESSIVE` The simplex optimizer should use an aggressive degeneration strategy.

`MSK_SIM_DEGEN_FREE` The simplex optimizer chooses the degeneration strategy.

Default value:
MSK_SIM_DEGEN_FREE

- `sim_dual_crash`

Corresponding constant:
`MSK_IPAR_SIM_DUAL_CRASH`

Description:
Controls whether crashing is performed in the dual simplex optimizer.

In general if a basis consists of more than (100-this parameter value)% fixed variables, then a crash will be performed.

Possible Values:
Any number between 0 and +inf.

Default value:
90

- `sim_dual_restrict_selection`

Corresponding constant:
`MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION`

Description:

The dual simplex optimizer can use a so-called restricted selection/pricing strategy to chooses the outgoing variable. Hence, if restricted selection is applied, then the dual simplex optimizer first choose a subset of all the potential outgoing variables. Next, for some time it will choose the outgoing variable only among the subset. From time to time the subset is redefined.

A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.

Possible Values:

Any number between 0 and 100.

Default value:

50

- `sim_dual_selection`

Corresponding constant:

`MSK_IPAR_SIM_DUAL_SELECTION`

Description:

Controls the choice of the incoming variable, known as the selection strategy, in the dual simplex optimizer.

Possible Values:

`MSK_SIM_SELECTION_FULL` The optimizer uses full pricing.

`MSK_SIM_SELECTION_PARTIAL` The optimizer uses a partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.

`MSK_SIM_SELECTION_FREE` The optimizer chooses the pricing strategy.

`MSK_SIM_SELECTION_ASE` The optimizer uses approximate steepest-edge pricing.

`MSK_SIM_SELECTION_DEVEX` The optimizer uses devex steepest-edge pricing (or if it is not available an approximate steep-edge selection).

`MSK_SIM_SELECTION_SE` The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

Default value:

MSK_SIM_SELECTION_FREE

- `sim_hotstart`

Corresponding constant:

`MSK_IPAR_SIM_HOTSTART`

Description:

Controls the type of hot-start that the simplex optimizer perform.

Possible Values:

`MSK_SIM_HOTSTART_NONE` The simplex optimizer performs a coldstart.

`MSK_SIM_HOTSTART_STATUS_KEYS` Only the status keys of the constraints and variables are used to choose the type of hot-start.

`MSK_SIM_HOTSTART_FREE` The simplex optimize chooses the hot-start type.

Default value:
MSK_SIM_HOTSTART_FREE

- `sim_max_iterations`

Corresponding constant:
`MSK_IPAR_SIM_MAX_ITERATIONS`

Description:
Maximum number of iterations that can be used by a simplex optimizer.

Possible Values:
Any number between 0 and +inf.

Default value:
10000000

- `sim_max_num_setbacks`

Corresponding constant:
`MSK_IPAR_SIM_MAX_NUM_SETBACKS`

Description:
Controls how many setbacks are allowed within a simplex optimizer. A setback is an event where the optimizer moves in the wrong direction. This is impossible in theory but may happen due to numerical problems.

Possible Values:
Any number between 0 and +inf.

Default value:
250

- `sim_network_detect`

Corresponding constant:
`MSK_IPAR_SIM_NETWORK_DETECT`

Description:
The simplex optimizer is capable of exploiting a network flow component in a problem. However it is only worthwhile to exploit the network flow component if it is sufficiently large. This parameter controls how large the network component has to be in “relative” terms before it is exploited. For instance a value of 20 means at least 20% of the model should be a network before it is exploited. If this value is larger than 100 the network flow component is never detected or exploited.

Possible Values:
Any number between 0 and +inf.

Default value:
101

- `sim_network_detect_hotstart`

  Corresponding constant:
  `MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART`

  Description:
  This parameter controls has large the network component in "relative" terms has to be before it is exploited in a simplex hot-start. The network component should be equal or larger than

  `max(MSK_IPAR_SIM_NETWORK_DETECT,MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART)`

  before it is exploited. If this value is larger than 100 the network flow component is never detected or exploited.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  100

- `sim_network_detect_method`

  Corresponding constant:
  `MSK_IPAR_SIM_NETWORK_DETECT_METHOD`

  Description:
  Controls which type of detection method the network extraction should use.

  Possible Values:

  `MSK_NETWORK_DETECT_SIMPLE` The network detection should use a very simple heuristic.

  `MSK_NETWORK_DETECT_ADVANCED` The network detection should use a more advanced heuristic.

  `MSK_NETWORK_DETECT_FREE` The network detection is free.

  Default value:
  MSK_NETWORK_DETECT_FREE

- `sim_non_singular`

  Corresponding constant:
  `MSK_IPAR_SIM_NON_SINGULAR`

  Description:
  Controls if the simplex optimizer ensures a non-singular basis, if possible.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `sim_primal_crash`

Corresponding constant:
`MSK_IPAR_SIM_PRIMAL_CRASH`

Description:
Controls whether crashing is performed in the primal simplex optimizer.

In general, if a basis consists of more than (100-this parameter value)% fixed variables, then a crash will be performed.

Possible Values:
Any nonnegative integer value.

Default value:
90

- `sim_primal_restrict_selection`

Corresponding constant:
`MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION`

Description:
The primal simplex optimizer can use a so-called restricted selection/pricing strategy to chooses the outgoing variable. Hence, if restricted selection is applied, then the primal simplex optimizer first choose a subset of all the potential incoming variables. Next, for some time it will choose the incoming variable only among the subset. From time to time the subset is redefined.

A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.

Possible Values:
Any number between 0 and 100.

Default value:
50

- `sim_primal_selection`

Corresponding constant:
`MSK_IPAR_SIM_PRIMAL_SELECTION`

Description:
Controls the choice of the incoming variable, known as the selection strategy, in the primal simplex optimizer.

Possible Values:

`MSK_SIM_SELECTION_FULL` The optimizer uses full pricing.

`MSK_SIM_SELECTION_PARTIAL` The optimizer uses a partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.

`MSK_SIM_SELECTION_FREE` The optimizer chooses the pricing strategy.

`MSK_SIM_SELECTION_ASE` The optimizer uses approximate steepest-edge pricing.

`MSK_SIM_SELECTION_DEVEX` The optimizer uses devex steepest-edge pricing (or if it is not available an approximate steep-edge selection).

`MSK_SIM_SELECTION_SE` The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

Default value:
MSK_SIM_SELECTION_FREE

- `sim_refactor_freq`

Corresponding constant:
`MSK_IPAR_SIM_REFACTOR_FREQ`

Description:
Controls how frequent the basis is refactorized. The value 0 means that the optimizer determines the best point of refactorization.

It is strongly recommended NOT to change this parameter.

Possible Values:
Any number between 0 and +inf.

Default value:
0

- `sim_save_lu`

Corresponding constant:
`MSK_IPAR_SIM_SAVE_LU`

Description:
Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `sim_scaling`

Corresponding constant:
`MSK_IPAR_SIM_SCALING`

Description:
Controls how the problem is scaled before a simplex optimizer is used.

Possible Values:

`MSK_SCALING_NONE` No scaling is performed.

`MSK_SCALING_MODERATE` A conservative scaling is performed.

`MSK_SCALING_AGGRESSIVE` A very aggressive scaling is performed.

`MSK_SCALING_FREE` The optimizer chooses the scaling heuristic.

Default value:
MSK_SCALING_FREE

- `sim_solve_form`

  Corresponding constant:
  `MSK_IPAR_SIM_SOLVE_FORM`

  Description:
  Controls whether the primal or the dual problem is solved by the primal-/dual- simplex optimizer.

  Possible Values:

  `MSK_SOLVE_PRIMAL` The optimizer should solve the primal problem.

  `MSK_SOLVE_DUAL` The optimizer should solve the dual problem.

  `MSK_SOLVE_FREE` The optimizer is free to solve either the primal or the dual problem.

  Default value:
  MSK_SOLVE_FREE

- `sim_stability_priority`

  Corresponding constant:
  `MSK_IPAR_SIM_STABILITY_PRIORITY`

  Description:
  Controls how high priority the numerical stability should be given.

  Possible Values:
  Any number between 0 and 100.

  Default value:
  50

- `sim_switch_optimizer`

  Corresponding constant:
  `MSK_IPAR_SIM_SWITCH_OPTIMIZER`

  Description:
  The simplex optimizer sometimes chooses to solve the dual problem instead of the primal problem. This implies that if you have chosen to use the dual simplex optimizer and the problem is dualized, then it actually makes sense to use the primal simplex optimizer instead. If this parameter is on and the problem is dualized and furthermore the simplex optimizer is chosen to be the primal (dual) one, then it is switched to the dual (primal).

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `sol_filter_keep_basic`

Corresponding constant:
`MSK_IPAR_SOL_FILTER_KEEP_BASIC`

Description:
If turned on, then basic and super basic constraints and variables are written to the solution file independent of the filter setting.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `sol_filter_keep_ranged`

Corresponding constant:
`MSK_IPAR_SOL_FILTER_KEEP_RANGED`

Description:
If turned on, then ranged constraints and variables are written to the solution file independent of the filter setting.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `sol_quoted_names`

Corresponding constant:
`MSK_IPAR_SOL_QUOTED_NAMES`

Description:
If this options is turned on, then MOSEK will quote names that contains blanks while writing the solution file. Moreover when reading leading and trailing quotes will be stripped of.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `sol_read_name_width`

Corresponding constant:
`MSK_IPAR_SOL_READ_NAME_WIDTH`

Description:
When a solution is read by MOSEK and some constraint, variable or cone names contain blanks, then a maximum name width much be specified. A negative value implies that no name contain blanks.

Possible Values:
Any number between -inf and +inf.

Default value:
-1

- `sol_read_width`

  Corresponding constant:
  `MSK_IPAR_SOL_READ_WIDTH`

  Description:
  Controls the maximal acceptable width of line in the solutions when read by MOSEK.

  Possible Values:
  Any positive number greater than 80.

  Default value:
  1024

- `solution_callback`

  Corresponding constant:
  `MSK_IPAR_SOLUTION_CALLBACK`

  Description:
  Indicates whether solution call-backs will be performed during the optimization.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `warning_level`

  Corresponding constant:
  `MSK_IPAR_WARNING_LEVEL`

  Description:
  Warning level.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  1

- `write_bas_constraints`

Corresponding constant:
`MSK_IPAR_WRITE_BAS_CONSTRAINTS`

Description:
Controls whether the constraint section is written to the basic solution file.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `write_bas_head`

Corresponding constant:
`MSK_IPAR_WRITE_BAS_HEAD`

Description:
Controls whether the header section is written to the basic solution file.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `write_bas_variables`

Corresponding constant:
`MSK_IPAR_WRITE_BAS_VARIABLES`

Description:
Controls whether the variables section is written to the basic solution file.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `write_data_compressed`

Corresponding constant:
`MSK_IPAR_WRITE_DATA_COMPRESSED`

Description:
Controls whether the data file is compressed while it is written. 0 means no compression while higher values mean more compression.

Possible Values:
Any number between 0 and +inf.

Default value:
0

- `write_data_format`

  Corresponding constant:
  `MSK_IPAR_WRITE_DATA_FORMAT`

  Description:

  Controls the file format when writing task data to a file.

  Possible Values:

  `MSK_DATA_FORMAT_XML` The data file is an XML formatted file.

  `MSK_DATA_FORMAT_EXTENSION` The file extension is used to determine the data file format.

  `MSK_DATA_FORMAT_MPS` The data file is MPS formatted.

  `MSK_DATA_FORMAT_LP` The data file is LP formatted.

  `MSK_DATA_FORMAT_MBT` The data file is a MOSEK binary task file.

  `MSK_DATA_FORMAT_OP` The data file is an optimization problem formatted file.

  Default value:
  MSK_DATA_FORMAT_EXTENSION

- `write_data_param`

  Corresponding constant:
  `MSK_IPAR_WRITE_DATA_PARAM`

  Description:
  If this option is turned on the parameter settings are written to the data file as parameters.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `write_free_con`

  Corresponding constant:
  `MSK_IPAR_WRITE_FREE_CON`

  Description:
  Controls whether the free constraints are written to the data file.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `write_generic_names`

  Corresponding constant:
  `MSK_IPAR_WRITE_GENERIC_NAMES`

  Description:
  Controls whether the generic names or user-defined names are used in the data file.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `write_generic_names_io`

  Corresponding constant:
  `MSK_IPAR_WRITE_GENERIC_NAMES_IO`

  Description:
  Index origin used in generic names.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  1

- `write_int_constraints`

  Corresponding constant:
  `MSK_IPAR_WRITE_INT_CONSTRAINTS`

  Description:
  Controls whether the constraint section is written to the integer solution file.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `write_int_head`

  Corresponding constant:
  `MSK_IPAR_WRITE_INT_HEAD`

  Description:
  Controls whether the header section is written to the integer solution file.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `write_int_variables`

  Corresponding constant:
  `MSK_IPAR_WRITE_INT_VARIABLES`

  Description:
  Controls whether the variables section is written to the integer solution file.

  Possible Values:

  `MSK_ON` Switch the option on.
  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `write_lp_line_width`

  Corresponding constant:
  `MSK_IPAR_WRITE_LP_LINE_WIDTH`

  Description:
  Maximum width of line in an LP file written by MOSEK.

  Possible Values:
  Any positive number.

  Default value:
  80

- `write_lp_quoted_names`

  Corresponding constant:
  `MSK_IPAR_WRITE_LP_QUOTED_NAMES`

  Description:
  If this option is turned on, then MOSEK will quote invalid LP names when writing an LP file.

  Possible Values:

  `MSK_ON` Switch the option on.
  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `write_lp_strict_format`

  Corresponding constant:
  `MSK_IPAR_WRITE_LP_STRICT_FORMAT`

  Description:
  Controls whether LP output files satisfy the LP format strictly.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `write_lp_terms_per_line`

  Corresponding constant:
  `MSK_IPAR_WRITE_LP_TERMS_PER_LINE`

  Description:
  Maximum number of terms on a single line in an LP file written by MOSEK. 0 means unlimited.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  10

- `write_mps_int`

  Corresponding constant:
  `MSK_IPAR_WRITE_MPS_INT`

  Description:
  Controls if marker records are written to the MPS file to indicate whether variables are integer restricted.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `write_mps_obj_sense`

  Corresponding constant:
  `MSK_IPAR_WRITE_MPS_OBJ_SENSE`

  Description:
  If turned off, the objective sense section is not written to the MPS file.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `write_mps_quoted_names`

Corresponding constant:
`MSK_IPAR_WRITE_MPS_QUOTED_NAMES`

Description:
If a name contains spaces (blanks) when writing an MPS file, then the quotes will be removed.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `write_mps_strict`

Corresponding constant:
`MSK_IPAR_WRITE_MPS_STRICT`

Description:
Controls whether the written MPS file satisfies the MPS format strictly or not.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `write_precision`

Corresponding constant:
`MSK_IPAR_WRITE_PRECISION`

Description:
Controls the precision with which `double` numbers are printed in the MPS data file. In general it is not worthwhile to use a value higher than 15.

Possible Values:
Any number between 0 and +inf.

Default value:
8

- `write_sol_constraints`

Corresponding constant:
`MSK_IPAR_WRITE_SOL_CONSTRAINTS`

Description:
Controls whether the constraint section is written to the solution file.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `write_sol_head`

  Corresponding constant:
  `MSK_IPAR_WRITE_SOL_HEAD`

  Description:
  Controls whether the header section is written to the solution file.

  Possible Values:

  `MSK_ON` Switch the option on.
  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `write_sol_variables`

  Corresponding constant:
  `MSK_IPAR_WRITE_SOL_VARIABLES`

  Description:
  Controls whether the variables section is written to the solution file.

  Possible Values:

  `MSK_ON` Switch the option on.
  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `write_task_inc_sol`

  Corresponding constant:
  `MSK_IPAR_WRITE_TASK_INC_SOL`

  Description:
  Controls whether the solutions are stored in the task file too.

  Possible Values:

  `MSK_ON` Switch the option on.
  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `write_xml_mode`

  Corresponding constant:
  `MSK_IPAR_WRITE_XML_MODE`

Description:

Controls if linear coefficients should be written by row or column when writing in the XML file format.

Possible Values:

`MSK_WRITE_XML_MODE_COL`	Write in column order.
`MSK_WRITE_XML_MODE_ROW`	Write in row order.

Default value:

MSK_WRITE_XML_MODE_ROW

H.4 String parameter types

- `MSK_SPAR_BAS_SOL_FILE_NAME` 246
  Name of the bas solution file.

- `MSK_SPAR_DATA_FILE_NAME` 246
  Data are read and written to this file.

- `MSK_SPAR_DEBUG_FILE_NAME` 247
  MOSEK debug file.

- `MSK_SPAR_FEASREPAIR_NAME_PREFIX` 247
  Feasibility repair name prefix.

- `MSK_SPAR_FEASREPAIR_NAME_SEPARATOR` 247
  Feasibility repair name separator.

- `MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL` 247
  Feasibility repair name violation name.

- `MSK_SPAR_INT_SOL_FILE_NAME` 248
  Name of the int solution file.

- `MSK_SPAR_ITR_SOL_FILE_NAME` 248
  Name of the itr solution file.

- `MSK_SPAR_PARAM_COMMENT_SIGN` 248
  Solution file comment character.

- `MSK_SPAR_PARAM_READ_FILE_NAME` 248
  Modifications to the parameter database is read from this file.

- `MSK_SPAR_PARAM_WRITE_FILE_NAME` 249
  The parameter database is written to this file.

- `MSK_SPAR_READ_MPS_BOU_NAME` 249
  Name of the BOUNDS vector used. An empty name means that the first BOUNDS vector is used.

- MSK_SPAR_READ_MPS_OBJ_NAME 249
  Objective name in the MPS file.
- MSK_SPAR_READ_MPS_RAN_NAME 249
  Name of the RANGE vector used. An empty name means that the first RANGE vector is used.
- MSK_SPAR_READ_MPS_RHS_NAME 250
  Name of the RHS used. An empty name means that the first RHS vector is used.
- MSK_SPAR_SENSITIVITY_FILE_NAME 250
  Sensitivity report file name.
- MSK_SPAR_SENSITIVITY_RES_FILE_NAME 250
  Name of the sensitivity report output file.
- MSK_SPAR_SOL_FILTER_XC_LOW 251
  Solution file filter.
- MSK_SPAR_SOL_FILTER_XC_UPR 251
  Solution file filter.
- MSK_SPAR_SOL_FILTER_XX_LOW 251
  Solution file filter.
- MSK_SPAR_SOL_FILTER_XX_UPR 251
  Solution file filter.
- MSK_SPAR_STAT_FILE_NAME 252
  Statistics file name.
- MSK_SPAR_STAT_KEY 252
  Key used when writing the summary file.
- MSK_SPAR_STAT_NAME 252
  Name used when writing the statistics file.
- MSK_SPAR_WRITE_LP_GEN_VAR_NAME 252
  Added variable names in the LP files.

- `bas_sol_file_name`

  Corresponding constant:
  `MSK_SPAR_BAS_SOL_FILE_NAME`

  Description:
  Name of the `bas` solution file.

  Possible Values:
  Any valid file name.

  Default value:
  ""

- `data_file_name`

Corresponding constant:
`MSK_SPAR_DATA_FILE_NAME`

Description:
Data are read and written to this file.

Possible Values:
Any valid file name.

Default value:
""

- `debug_file_name`

  Corresponding constant:
  `MSK_SPAR_DEBUG_FILE_NAME`

  Description:
  MOSEK debug file.

  Possible Values:
  Any valid file name.

  Default value:
  ""

- `feasrepair_name_prefix`

  Corresponding constant:
  `MSK_SPAR_FEASREPAIR_NAME_PREFIX`

  Description:

  Not applicable.

  Possible Values:
  Any valid string.

  Default value:
  "MSK-"

- `feasrepair_name_separator`

  Corresponding constant:
  `MSK_SPAR_FEASREPAIR_NAME_SEPARATOR`

  Description:

  Not applicable.

  Possible Values:
  Any valid string.

  Default value:
  "-"

- `feasrepair_name_wsumviol`

Corresponding constant:
`MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL`

Description:
The constraint and variable associated with the total weighted sum of violations are each given the name of this parameter postfixed with `CON` and `VAR` respectively.

Possible Values:
Any valid string.

Default value:
"WSUMVIOL"

- `int_sol_file_name`

Corresponding constant:
`MSK_SPAR_INT_SOL_FILE_NAME`

Description:
Name of the `int` solution file.

Possible Values:
Any valid file name.

Default value:
""

- `itr_sol_file_name`

Corresponding constant:
`MSK_SPAR_ITR_SOL_FILE_NAME`

Description:
Name of the `itr` solution file.

Possible Values:
Any valid file name.

Default value:
""

- `param_comment_sign`

Corresponding constant:
`MSK_SPAR_PARAM_COMMENT_SIGN`

Description:
Only the first character in this string is used. It is considered as a start of comment sign in the MOSEK parameter file. Spaces are ignored in the string.

Possible Values:
Any valid string.

Default value:
"%%"

- `param_read_file_name`

Corresponding constant:
`MSK_SPAR_PARAM_READ_FILE_NAME`

Description:
Modifications to the parameter database is read from this file.

Possible Values:
Any valid file name.

Default value:
””

- `param_write_file_name`

  Corresponding constant:
  `MSK_SPAR_PARAM_WRITE_FILE_NAME`

  Description:
  The parameter database is written to this file.

  Possible Values:
  Any valid file name.

  Default value:
  ””

- `read_mps_bou_name`

  Corresponding constant:
  `MSK_SPAR_READ_MPS_BOU_NAME`

  Description:
  Name of the BOUNDS vector used. An empty name means that the first BOUNDS vector is used.

  Possible Values:
  Any valid MPS name.

  Default value:
  ””

- `read_mps_obj_name`

  Corresponding constant:
  `MSK_SPAR_READ_MPS_OBJ_NAME`

  Description:
  Name of the free constraint used as objective function. An empty name means that the first constraint is used as objective function.

  Possible Values:
  Any valid MPS name.

  Default value:
  ””

- `read_mps_ran_name`

Corresponding constant:
`MSK_SPAR_READ_MPS_RAN_NAME`

Description:
Name of the RANGE vector used. An empty name means that the first RANGE vector is used.

Possible Values:
Any valid MPS name.

Default value:
""

- `read_mps_rhs_name`

Corresponding constant:
`MSK_SPAR_READ_MPS_RHS_NAME`

Description:
Name of the RHS used. An empty name means that the first RHS vector is used.

Possible Values:
Any valid MPS name.

Default value:
""

- `sensitivity_file_name`

Corresponding constant:
`MSK_SPAR_SENSITIVITY_FILE_NAME`

Description:

Not applicable.

Possible Values:
Any valid string.

Default value:
""

- `sensitivity_res_file_name`

Corresponding constant:
`MSK_SPAR_SENSITIVITY_RES_FILE_NAME`

Description:

Not applicable.

Possible Values:
Any valid string.

Default value:
""

- `sol_filter_xc_low`

  Corresponding constant:
  `MSK_SPAR_SOL_FILTER_XC_LOW`

  Description:
  A filter used to determine which constraints should be listed in the solution file. A value of "0.5" means that all constraints having `xc[i]>0.5` should be listed, whereas "+0.5" means that all constraints having `xc[i]>=blc[i]+0.5` should be listed. An empty filter means that no filter is applied.

  Possible Values:
  Any valid filter.

  Default value:
  ""

- `sol_filter_xc_upr`

  Corresponding constant:
  `MSK_SPAR_SOL_FILTER_XC_UPR`

  Description:
  A filter used to determine which constraints should be listed in the solution file. A value of "0.5" means that all constraints having `xc[i]<0.5` should be listed, whereas "-0.5" means all constraints having `xc[i]<=buc[i]-0.5` should be listed. An empty filter means that no filter is applied.

  Possible Values:
  Any valid filter.

  Default value:
  ""

- `sol_filter_xx_low`

  Corresponding constant:
  `MSK_SPAR_SOL_FILTER_XX_LOW`

  Description:
  A filter used to determine which variables should be listed in the solution file. A value of "0.5" means that all constraints having `xx[j]>=0.5` should be listed, whereas "+0.5" means that all constraints having `xx[j]>=blx[j]+0.5` should be listed. An empty filter means no filter is applied.

  Possible Values:
  Any valid filter..

  Default value:
  ""

- `sol_filter_xx_upr`

  Corresponding constant:
  `MSK_SPAR_SOL_FILTER_XX_UPR`

Description:
A filter used to determine which variables should be listed in the solution file. A value of "0.5" means that all constraints having `xx[j]<0.5` should be printed, whereas "-0.5" means all constraints having `xx[j]<=bux[j]-0.5` should be listed. An empty filter means no filter is applied.

Possible Values:
Any valid file name.

Default value:
""

- `stat_file_name`

  Corresponding constant:
  `MSK_SPAR_STAT_FILE_NAME`

  Description:
  Statistics file name.

  Possible Values:
  Any valid file name.

  Default value:
  ""

- `stat_key`

  Corresponding constant:
  `MSK_SPAR_STAT_KEY`

  Description:
  Key used when writing the summary file.

  Possible Values:
  Any valid XML string.

  Default value:
  ""

- `stat_name`

  Corresponding constant:
  `MSK_SPAR_STAT_NAME`

  Description:
  Name used when writing the statistics file.

  Possible Values:
  Any valid XML string.

  Default value:
  ""

- `write_lp_gen_var_name`

Corresponding constant:

`MSK_SPAR_WRITE_LP_GEN_VAR_NAME`

Description:

Sometimes when an LP file is written additional variables must be inserted. They will have the prefix denoted by this parameter.

Possible Values:

Any valid string.

Default value:

”xmskgen”

Appendix I

Symbolic constants reference

I.1 Constraint or variable access modes

Value	Name Description
0	`MSK_ACC_VAR` Access data by columns (variable orinted)
1	`MSK_ACC_CON` Access data by rows (constraint oriented)

I.2 Basis identification

Value	Name Description
1	`MSK_BI_ALWAYS` Basis identification is always performed even if the interior-point optimizer terminates abnormally.
2	`MSK_BI_NO_ERROR` Basis identification is performed if the interior-point optimizer terminates without an error.
0	`MSK_BI_NEVER` Never do basis identification.
3	`MSK_BI_IF_FEASIBLE` Basis identification is not performed if the interior-point optimizer terminates with a problem status saying that the problem is primal or dual infeasible.
4	`MSK_BI_OTHER` Try another BI method.

I.3 Bound keys

Value	Name	Description
2	`MSK_BK_FX`	The constraint or variable is fixed.
0	`MSK_BK_LO`	The constraint or variable has a finite lower bound and an infinite upper bound.
3	`MSK_BK_FR`	The constraint or variable is free.
1	`MSK_BK_UP`	The constraint or variable has an infinite lower bound and an finite upper bound.
4	`MSK_BK_RA`	The constraint or variable is ranged.

I.4 Specifies the branching direction.

Value	Name	Description
2	`MSK_BRANCH_DIR_DOWN`	The mixed integer optimizer always chooses the down branch first.
1	`MSK_BRANCH_DIR_UP`	The mixed integer optimizer always chooses the up branch first.
0	`MSK_BRANCH_DIR_FREE`	The mixed optimizer decides which branch to choose.

I.5 Progress call-back codes

Value	Name	Description
17	`MSK_CALLBACK_BEGIN_PRIMAL_SENSITIVITY`	Primal sensitivity analysis is started.
70	`MSK_CALLBACK_NEW_INT_MIO`	The call-back function is called after a new integer solution has been located by the mixed integer optimizer.
37	`MSK_CALLBACK_END_NETWORK_PRIMAL_SIMPLEX`	The call-back function is called when the primal network simplex optimizer is terminated.
79	`MSK_CALLBACK_UPDATE_PRESOLVE`	

continued on next page

continued from previous page

	The call-back function is called from within the presolve procedure.
55	`MSK_CALLBACK_IM_LICENSE_WAIT` MOSEK is waiting for a license.
1	`MSK_CALLBACK_BEGIN_CONCURRENT` Concurrent optimizer is started.
76	`MSK_CALLBACK_UPDATE_NETWORK_DUAL_SIMPLEX` The call-back function is called in the dual network simplex optimizer.
48	`MSK_CALLBACK_IGNORE_VALUE` This code means that the call-back does not indicate a new phase in the optimization, but is simply a time-triggered call-back.
46	`MSK_CALLBACK_END_SIMPLEX_BI` The call-back function is called from within the basis identification procedure when the simplex clean-up phase is terminated.
42	`MSK_CALLBACK_END_PRIMAL_SENSITIVITY` Primal sensitivity analysis is terminated.
20	`MSK_CALLBACK_BEGIN_SIMPLEX` The call-back function is called when the simplex optimizer is started.
40	`MSK_CALLBACK_END_PRESOLVE` The call-back function is called when the presolve is completed.
22	`MSK_CALLBACK_BEGIN_SIMPLEX_NETWORK_DETECT` The call-back function is called when the network detection procedure is started.
13	`MSK_CALLBACK_BEGIN_NETWORK_SIMPLEX` The call-back function is called when the simplex network optimizer is started.
35	`MSK_CALLBACK_END_MIO` The call-back function is called when the mixed integer optimizer is terminated.
73	`MSK_CALLBACK_QCONE` The call-back function is called from within the Qcone optimizer.
27	`MSK_CALLBACK_END_CONIC` The call-back function is called when the conic optimizer is terminated.
11	`MSK_CALLBACK_BEGIN_NETWORK_DUAL_SIMPLEX` The call-back function is called when the dual network simplex optimizer is started.
7	`MSK_CALLBACK_BEGIN_INFEAS_ANA` The call-back function is called when the infeasibility analyzer is started.
67	`MSK_CALLBACK_IM_PRIMAL_SIMPLEX` The call-back function is called at an intermediate point in the primal simplex optimizer.
63	`MSK_CALLBACK_IM_NONCONVEX`

continued on next page

continued from previous page

	The call-back function is called at an intermediate stage within the nonconvex optimizer where the information database has not been updated.
61	`MSK_CALLBACK_IM_NETWORK_DUAL_SIMPLEX` The call-back function is called at an intermediate point in the dual network simplex optimizer.
66	`MSK_CALLBACK_IM_PRIMAL_SENSIVITY` The call-back function is called at an intermediate stage of the primal sensitivity analysis.
38	`MSK_CALLBACK_END_NETWORK_SIMPLEX` The call-back function is called when the simplex network optimizer is terminated.
34	`MSK_CALLBACK_END_LICENSE_WAIT` End waiting for license.
28	`MSK_CALLBACK_END_DUAL_BI` The call-back function is called from within the basis identification procedure when the dual phase is terminated.
29	`MSK_CALLBACK_END_DUAL_SENSITIVITY` Dual sensitivity analysis is terminated.
24	`MSK_CALLBACK_DUAL_SIMPLEX` The call-back function is called from within the dual simplex optimizer.
21	`MSK_CALLBACK_BEGIN_SIMPLEX_BI` The call-back function is called from within the basis identification procedure when the simplex clean-up phase is started.
8	`MSK_CALLBACK_BEGIN_INTPNT` The call-back function is called when the interior-point optimizer is started.
52	`MSK_CALLBACK_IM_DUAL_SENSIVITY` The call-back function is called at an intermediate stage of the dual sensitivity analysis.
47	`MSK_CALLBACK_END_SIMPLEX_NETWORK_DETECT` The call-back function is called when the network detection procedure is terminated.
45	`MSK_CALLBACK_END_SIMPLEX` The call-back function is called when the simplex optimizer is terminated.
72	`MSK_CALLBACK_PRIMAL_SIMPLEX` The call-back function is called from within the primal simplex optimizer.
26	`MSK_CALLBACK_END_CONCURRENT` Concurrent optimizer is terminated.
56	`MSK_CALLBACK_IM_MIO`

continued on next page

continued from previous page

	The call-back function is called at an intermediate point in the mixed integer optimizer.
31	`MSK_CALLBACK_END_DUAL_SIMPLEX` The call-back function is called when the dual simplex optimizer is terminated.
36	`MSK_CALLBACK_END_NETWORK_DUAL_SIMPLEX` The call-back function is called when the dual network simplex optimizer is terminated.
54	`MSK_CALLBACK_IM_INTPNT` The call-back function is called at an intermediate stage within the interior-point optimizer where the information database has not been updated.
68	`MSK_CALLBACK_IM_SIMPLEX_BI` The call-back function is called from within the basis identification procedure at an intermediate point in the simplex clean-up phase. The frequency of the call-backs is controlled by the `MSK_IPAR_LOG_SIM_FREQ` parameter.
51	`MSK_CALLBACK_IM_DUAL_BI` The call-back function is called from within the basis identification procedure at an intermediate point in the dual phase.
75	`MSK_CALLBACK_UPDATE_DUAL_SIMPLEX` The call-back function is called in the dual simplex optimizer.
82	`MSK_CALLBACK_UPDATE_SIMPLEX_BI` The call-back function is called from within the basis identification procedure at an intermediate point in the simplex clean-up phase. The frequency of the call-backs is controlled by the `MSK_IPAR_LOG_SIM_FREQ` parameter.
19	`MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX` The call-back function is called when the primal simplex optimizer is started.
58	`MSK_CALLBACK_IM_MIO_INTPNT` The call-back function is called at an intermediate point in the mixed integer optimizer while running the interior-point optimizer.
6	`MSK_CALLBACK_BEGIN_DUAL_SIMPLEX` The call-back function is called when the dual simplex optimizer started.
64	`MSK_CALLBACK_IM_PRESOLVE` The call-back function is called from within the presolve procedure at an intermediate stage.
30	`MSK_CALLBACK_END_DUAL_SETUP_BI` The call-back function is called when the dual BI phase is terminated.
3	`MSK_CALLBACK_BEGIN_DUAL_BI` The call-back function is called from within the basis identification procedure when the dual phase is started.

continued on next page

continued from previous page

4	`MSK_CALLBACK_BEGIN_DUAL_SENSITIVITY` Dual sensitivity analysis is started.
50	`MSK_CALLBACK_IM_CONIC` The call-back function is called at an intermediate stage within the conic optimizer where the information database has not been updated.
60	`MSK_CALLBACK_IM_MIO_PRIMAL_SIMPLEX` The call-back function is called at an intermediate point in the mixed integer optimizer while running the primal simplex optimizer.
77	`MSK_CALLBACK_UPDATE_NETWORK_PRIMAL_SIMPLEX` The call-back function is called in the primal network simplex optimizer.
59	`MSK_CALLBACK_IM_MIO_PRESOLVE` The call-back function is called at an intermediate point in the mixed integer optimizer while running the presolve.
14	`MSK_CALLBACK_BEGIN_NONCONVEX` The call-back function is called when the nonconvex optimizer is started.
0	`MSK_CALLBACK_BEGIN_BI` The basis identification procedure has been started.
33	`MSK_CALLBACK_END_INTPNT` The call-back function is called when the interior-point optimizer is terminated.
16	`MSK_CALLBACK_BEGIN_PRIMAL_BI` The call-back function is called from within the basis identification procedure when the primal phase is started.
41	`MSK_CALLBACK_END_PRIMAL_BI` The call-back function is called from within the basis identification procedure when the primal phase is terminated.
18	`MSK_CALLBACK_BEGIN_PRIMAL_SETUP_BI` The call-back function is called when the primal BI setup is started.
32	`MSK_CALLBACK_END_INFEAS_ANA` The call-back function is called when the infeasibility analyzer is terminated.
74	`MSK_CALLBACK_UPDATE_DUAL_BI` The call-back function is called from within the basis identification procedure at an intermediate point in the dual phase.
39	`MSK_CALLBACK_END_NONCONVEX` The call-back function is called when the nonconvex optimizer is terminated.
69	`MSK_CALLBACK_INTPNT` The call-back function is called from within the interior-point optimizer after the information database has been updated.
53	`MSK_CALLBACK_IM_DUAL_SIMPLEX`

continued on next page

continued from previous page

	The call-back function is called at an intermediate point in the dual simplex optimizer.
44	`MSK_CALLBACK_END_PRIMAL_SIMPLEX` The call-back function is called when the primal simplex optimizer is terminated.
81	`MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX` The call-back function is called in the primal simplex optimizer.
80	`MSK_CALLBACK_UPDATE_PRIMAL_BI` The call-back function is called from within the basis identification procedure at an intermediate point in the primal phase.
71	`MSK_CALLBACK_NONCOVEX` The call-back function is called from within the nonconvex optimizer after the information database has been updated.
62	`MSK_CALLBACK_IM_NETWORK_PRIMAL_SIMPLEX` The call-back function is called at an intermediate point in the primal network simplex optimizer.
65	`MSK_CALLBACK_IM_PRIMAL_BI` The call-back function is called from within the basis identification procedure at an intermediate point in the primal phase.
57	`MSK_CALLBACK_IM_MIO_DUAL_SIMPLEX` The call-back function is called at an intermediate point in the mixed integer optimizer while running the dual simplex optimizer.
15	`MSK_CALLBACK_BEGIN_PRESOLVE` The call-back function is called when the presolve is started.
23	`MSK_CALLBACK_CONIC` The call-back function is called from within the conic optimizer after the information database has been updated.
49	`MSK_CALLBACK_IM_BI` The call-back function is called from within the basis identification procedure at an intermediate point.
43	`MSK_CALLBACK_END_PRIMAL_SETUP_BI` The call-back function is called when the primal BI setup is terminated.
10	`MSK_CALLBACK_BEGIN_MIO` The call-back function is called when the mixed integer optimizer is started.
12	`MSK_CALLBACK_BEGIN_NETWORK_PRIMAL_SIMPLEX` The call-back function is called when the primal network simplex optimizer is started.
2	`MSK_CALLBACK_BEGIN_CONIC` The call-back function is called when the conic optimizer is started.
9	`MSK_CALLBACK_BEGIN_LICENSE_WAIT` Begin waiting for license.
25	`MSK_CALLBACK_END_BI`

continued on next page

continued from previous page

	The call-back function is called when the basis identification procedure is terminated.
78	`MSK_CALLBACK_UPDATE_NONCONVEX`
	The call-back function is called at an intermediate stage within the nonconvex optimizer where the information database has been updated.
5	`MSK_CALLBACK_BEGIN_DUAL_SETUP_BI`
	The call-back function is called when the dual BI phase is started.

I.6 Types of convexity checks.

Value	Name Description
1	`MSK_CHECK_CONVEXITY_SIMPLE`
	Perform simple and fast convexity check.
0	`MSK_CHECK_CONVEXITY_NONE`
	No convexity check.

I.7 Compression types

Value	Name Description
2	`MSK_COMPRESS_GZIP`
	The type of compression used is gzip compatible.
0	`MSK_COMPRESS_NONE`
	No compression is used.
1	`MSK_COMPRESS_FREE`
	The type of compression used is chosen automatically.

I.8 Cone types

Value	Name Description
0	`MSK_CT_QUAD`
	The cone is a quadratic cone.
1	`MSK_CT_RQUAD`
	The cone is a rotated quadratic cone.

I.9 CPU type

Value	Name Description
8	`MSK_CPU_POWERPC_G5` A G5 PowerPC CPU.
9	`MSK_CPU_INTEL_PM` An Intel PM cpu.
1	`MSK_CPU_GENERIC` An generic CPU type for the platform
0	`MSK_CPU_UNKNOWN` An unknown CPU.
7	`MSK_CPU_AMD_OPTERON` An AMD Opteron (64 bit).
6	`MSK_CPU_INTEL_ITANIUM2` An Intel Itanium2.
4	`MSK_CPU_AMD_ATHLON` An AMD Athlon.
5	`MSK_CPU_HP_PARISC20` An HP PA RISC version 2.0 CPU.
3	`MSK_CPU_INTEL_P4` An Intel Pentium P4 or Intel Xeon.
2	`MSK_CPU_INTEL_P3` An Intel Pentium P3.
10	`MSK_CPU_INTEL_CORE2` An Intel CORE2 cpu.

I.10 Data format types

Value	Name Description
5	`MSK_DATA_FORMAT_XML` The data file is an XML formatted file.
0	`MSK_DATA_FORMAT_EXTENSION` The file extension is used to determine the data file format.
1	`MSK_DATA_FORMAT_MPS` The data file is MPS formatted.
2	`MSK_DATA_FORMAT_LP` The data file is LP formatted.
3	`MSK_DATA_FORMAT_MBT` The data file is a MOSEK binary task file.
4	`MSK_DATA_FORMAT_OP` The data file is an optimization problem formatted file.

I.11 Double information items

Value	Name	Description
12	`MSK_DINF_INTPNT_PRIMAL_FEAS`	Primal feasibility measure reported by the interior-point or Qcone optimizers. (For the interior-point optimizer this measure does not directly related to the original problem because a homogeneous model is employed).
11	`MSK_DINF_INTPNT_ORDER_CPUTIME`	Order time (in CPU seconds).
24	`MSK_DINF_PRESOLVE_CPUTIME`	Total time (in CPU seconds) spent in the presolve since it was invoked.
27	`MSK_DINF_RD_CPUTIME`	Time (in CPU seconds) spent reading the data file.
28	`MSK_DINF_SIM_CPUTIME`	Time (in CPU seconds) spent in the simplex optimizer since invoking it.
32	`MSK_DINF_SOL_BAS_MAX_DBI`	Maximal dual bound infeasibility in the basic solution. Updated at the end of the optimization.
47	`MSK_DINF_SOL_ITR_MAX_PCNI`	Maximal primal cone infeasibility in the interior-point solution. Updated at the end of the optimization.
19	`MSK_DINF_MIO_OBJ_INT`	The primal objective value corresponding to the best integer feasible solution. Please note that at least one integer feasible solution must have located i.e. check `MSK_IINF_MIO_NUM_INT_SOLUTIONS`.
4	`MSK_DINF_CONCURRENT_CPUTIME`	Time (in CPU seconds) spent within the concurrent optimizer since its invocation.
49	`MSK_DINF_SOL_ITR_MAX_PINTI`	Maximal primal integer infeasibility in the interior-point solution. Updated at the end of the optimization.
30	`MSK_DINF_SIM_OBJ`	Objective value reported by the simplex optimizer.
20	`MSK_DINF_MIO_OBJ_REL_GAP`	Given that the mixed integer optimizer has computed a feasible solution and a bound on the optimal objective value, then this item contains the relative gap defined by $\frac{\lvert(\text{objective value of feasible solution}) - (\text{objective bound})\rvert}{\max(1, \lvert(\text{objective value of feasible solution})\rvert)}.$ Otherwise it has the value -1.0.
37	`MSK_DINF_SOL_BAS_PRIMAL_OBJ`	

continued on next page

continued from previous page

	Primal objective value of the basic solution. Updated at the end of the optimization.
26	`MSK_DINF_PRESOLVE_LINDEP_CPUTIME` Total time (in CPU seconds) spent in the linear dependency checker since the presolve was invoked.
46	`MSK_DINF_SOL_ITR_MAX_PBI` Maximal primal bound infeasibility in the interior-point solution. Updated at the end of the optimization.
34	`MSK_DINF_SOL_BAS_MAX_PBI` Maximal primal bound infeasibility in the basic solution. Updated at the end of the optimization.
44	`MSK_DINF_SOL_ITR_MAX_DCNI` Maximal dual cone infeasibility in the interior-point solution. Updated at the end of the optimization.
8	`MSK_DINF_INTPNT_DUAL_OBJ` Dual objective value reported by the interior-point or Qcone optimizer.
45	`MSK_DINF_SOL_ITR_MAX_DEQI` Maximal dual equality infeasibility in the interior-point solution. Updated at the end of the optimization.
14	`MSK_DINF_INTPNT_REALTIME` Time (in wall-clock seconds) spent within the interior-point optimizer since its invocation.
29	`MSK_DINF_SIM_FEAS` Feasibility measure reported by the simplex optimizer.
15	`MSK_DINF_MIO_CONSTRUCT_SOLUTION_OBJ` If MOSEK has successfully constructed an integer feasible solution, then this item contains the optimal objective value corresponding to the feasible solution.
36	`MSK_DINF_SOL_BAS_MAX_PINTI` Maximal primal integer infeasibility in the basic solution. Updated at the end of the optimization.
40	`MSK_DINF_SOL_INT_MAX_PINTI` Maximal primal integer infeasibility in the integer solution. Updated at the end of the optimization.
6	`MSK_DINF_INTPNT_CPUTIME` Time (in CPU seconds) spent within the interior-point optimizer since its invocation.
35	`MSK_DINF_SOL_BAS_MAX_PEQI` Maximal primal equality infeasibility in the basic solution. Updated at the end of the optimization.
9	`MSK_DINF_INTPNT_FACTOR_NUM_FLOPS` An estimate of the number of flops used in the factorization.
42	`MSK_DINF_SOL_ITR_DUAL_OBJ`

continued on next page

continued from previous page

	Dual objective value of the interior-point solution. Updated at the end of the optimization.
22	`MSK_DINF_OPTIMIZER_CPUTIME` Total time (in CPU seconds) spent in the optimizer since it was invoked.
38	`MSK_DINF_SOL_INT_MAX_PBI` Maximal primal bound infeasibility in the integer solution. Updated at the end of the optimization.
7	`MSK_DINF_INTPNT_DUAL_FEAS` Dual feasibility measure reported by the interior-point and Qcone optimizer. (For the interior-point optimizer this measure does not directly related to the original problem because a homogeneous model is employed.)
23	`MSK_DINF_OPTIMIZER_REALTIME` Total time (in wall-clock seconds) spent in the optimizer since it was invoked.
10	`MSK_DINF_INTPNT_KAP_DIV_TAU` This measure should converge to zero if the problem has a primal-dual optimal solution or to infinity if problem is (strictly) primal or dual infeasible. In case the measure is converging towards a positive but bounded constant the problem is usually ill-posed.
3	`MSK_DINF_BI_PRIMAL_CPUTIME` Time (in CPU seconds) spent within the primal phase of the basis identification procedure since its invocation.
39	`MSK_DINF_SOL_INT_MAX_PEQI` Maximal primal equality infeasibility in the basic solution. Updated at the end of the optimization.
50	`MSK_DINF_SOL_ITR_PRIMAL_OBJ` Primal objective value of the interior-point solution. Updated at the end of the optimization.
21	`MSK_DINF_MIO_USER_OBJ_CUT` If the objective cut is used, then this information item has the value of the cut.
25	`MSK_DINF_PRESOLVE_ELI_CPUTIME` Total time (in CPU seconds) spent in the eliminator since the presolve was invoked.
5	`MSK_DINF_CONCURRENT_REALTIME` Time (in wall-clock seconds) within the concurrent optimizer since its invocation.
18	`MSK_DINF_MIO_OBJ_BOUND` The best bound objective value corresponding to the best integer feasible solution is located. Please note that at least one integer feasible solution must be located i.e. check `MSK_IINF_MIO_NUM_INT_SOLUTIONS`.

continued on next page

continued from previous page

1	`MSK_DINF_BI_CPUTIME` Time (in CPU seconds) spent within the basis identification procedure since its invocation.
31	`MSK_DINF_SOL_BAS_DUAL_OBJ` Dual objective value of the basic solution. Updated at the end of the optimization.
13	`MSK_DINF_INTPNT_PRIMAL_OBJ` Primal objective value reported by the interior-point or Qcone optimizer.
41	`MSK_DINF_SOL_INT_PRIMAL_OBJ` Primal objective value of the integer solution. Updated at the end of the optimization.
16	`MSK_DINF_MIO_CPUTIME` Time spent in the mixed integer optimizer.
17	`MSK_DINF_MIO_OBJ_ABS_GAP` Given the mixed integer optimizer has computed a feasible solution and a bound on the optimal objective value, then this item contains the absolute gap defined by $\lvert(\text{objective value of feasible solution}) - (\text{objective bound})\rvert.$ Otherwise it has the value -1.0.
33	`MSK_DINF_SOL_BAS_MAX_DEQI` Maximal dual equality infeasibility in the basic solution. Updated at the end of the optimization.
48	`MSK_DINF_SOL_ITR_MAX_PEQI` Maximal primal equality infeasibility in the interior-point solution. Updated at the end of the optimization.
2	`MSK_DINF_BI_DUAL_CPUTIME` Time (in CPU seconds) spent within the dual phase basis identification procedure since its invocation.
0	`MSK_DINF_BI_CLEAN_CPUTIME` Time (in CPU seconds) spent within the clean-up phase of the basis identification procedure since its invocation.
43	`MSK_DINF_SOL_ITR_MAX_DBI` Maximal dual bound infeasibility in the interior-point solution. Updated at the end of the optimization.

I.12 Double parameters

Value	Name Description
38	`MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH`

continued on next page

continued from previous page

	If the lower objective cut is less than the value of this parameter value, then the lower objective cut i.e. MSK_DPAR_LOWER_OBJ_CUT is treated as $-\infty$.
41	`MSK_DPAR_MIO_MAX_TIME` This parameter limits the maximum time spent by the mixed integer optimizer. A negative number means infinity.
1	`MSK_DPAR_BASIS_TOL_S` Maximum absolute dual bound violation in an optimal basic solution.
56	`MSK_DPAR_PRESOLVE_TOL_S` Absolute zero tolerance employed for s_i in the presolve.
59	`MSK_DPAR_UPPER_OBJ_CUT` If a feasible solution having and objective value outside, the interval [MSK_DPAR_LOWER_OBJ_CUT, MSK_DPAR_UPPER_OBJ_CUT], then MOSEK is terminated.
14	`MSK_DPAR_INTPNT_CO_TOL_DFEAS` Dual feasibility tolerance used by the conic interior-point optimizer.
6	`MSK_DPAR_DATA_TOL_AIJ_LARGE` An element in A which is larger than this value in absolute size causes a warning message to be printed.
46	`MSK_DPAR_MIO_TOL_ABS_GAP` Absolute optimality tolerance employed by the mixed integer optimizer.
60	`MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH` If the upper objective cut is greater than the value of this value parameter, then the the upper objective cut MSK_DPAR_UPPER_OBJ_CUT is treated as ∞.
52	`MSK_DPAR_NONCONVEX_TOL_OPT` Optimality tolerance used by the nonconvex optimizer.
51	`MSK_DPAR_NONCONVEX_TOL_FEAS` Feasibility tolerance used by the nonconvex optimizer.
40	`MSK_DPAR_MIO_HEURISTIC_TIME` Minimum amount of time to be used in the heuristic search for a good feasible integer solution. A negative values implies that the optimizer decides the amount of time to be spent in the heuristic.
57	`MSK_DPAR_PRESOLVE_TOL_X` Absolute zero tolerance employed for x_j in the presolve.
22	`MSK_DPAR_INTPNT_NL_TOL_MU_RED` Relative complementarity gap tolerance.
44	`MSK_DPAR_MIO_NEAR_TOL_REL_GAP` The mixed integer optimizer is terminated when this tolerance is satisfied. This termination criteria is delayed. See MSK_DPAR_MIO_DISABLE_TERM_TIME for details.
58	`MSK_DPAR_SIMPLEX_ABS_TOL_PIV` Absolute pivot tolerance employed by the simplex optimizers.

continued on next page

continued from previous page

5	`MSK_DPAR_DATA_TOL_AIJ` Absolute zero tolerance for elements in A.
13	`MSK_DPAR_FEASREPAIR_TOL` Tolerance for constraint enforcing upper bound on sum of weighted violations in feasibility repair.
28	`MSK_DPAR_INTPNT_TOL_DSAFE` Controls the initial dual starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly.
29	`MSK_DPAR_INTPNT_TOL_INFEAS` Controls when the optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.
23	`MSK_DPAR_INTPNT_NL_TOL_NEAR_REL` If the MOSEK nonlinear interior-point optimizer cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.
53	`MSK_DPAR_OPTIMIZER_MAX_TIME` Maximum amount of time the optimizer is allowed to spent on the optimization. A negative number means infinity.
12	`MSK_DPAR_DATA_TOL_X` Zero tolerance for constraints and variables i.e. if the distance between the lower and upper bound is less than this value, then the lower and lower bound is considered identical.
45	`MSK_DPAR_MIO_REL_ADD_CUT_LIMITED` Controls how many cuts the mixed integer optimizer is allowed to add to the problem. Let α be the value of this parameter and m the number constraints, then mixed integer optimizer is allowed to αm cuts.
3	`MSK_DPAR_BI_LU_TOL_REL_PIV` Relative pivot tolerance used in the LU factorization in the basis identification procedure.
30	`MSK_DPAR_INTPNT_TOL_MU_RED` Relative complementarity gap tolerance.
16	`MSK_DPAR_INTPNT_CO_TOL_MU_RED` Relative complementarity gap tolerance feasibility tolerance used by the conic interior-point optimizer.
19	`MSK_DPAR_INTPNT_CO_TOL_REL_GAP` Relative gap termination tolerance used by the conic interior-point optimizer.
37	`MSK_DPAR_LOWER_OBJ_CUT`

continued on next page

continued from previous page

	If a feasible solution having an objective value outside, the interval [MSK_DPAR_LOWER_OBJ_CUT, MSK_DPAR_UPPER_OBJ_CUT], then MOSEK is terminated.
39	`MSK_DPAR_MIO_DISABLE_TERM_TIME` The termination criteria governed by • MSK_IPAR_MIO_MAX_NUM_RELAXS • MSK_IPAR_MIO_MAX_NUM_BRANCHES • MSK_DPAR_MIO_NEAR_TOL_ABS_GAP • MSK_DPAR_MIO_NEAR_TOL_REL_GAP is disabled the first n seconds. This parameter specifies the number n. A negative value is identical to infinity i.e. the termination criterias are never checked.
50	`MSK_DPAR_MIO_TOL_X` Absolute solution tolerance used in mixed-integer optimizer.
9	`MSK_DPAR_DATA_TOL_C_HUGE` An element in c which is larger than the value of this parameter in absolute terms is considered to be huge and generates an error.
55	`MSK_DPAR_PRESOLVE_TOL_LIN_DEP` Controls when a constraint is determined to be linearly dependent.
35	`MSK_DPAR_INTPNT_TOL_REL_STEP` Relative step size to the boundary for linear and quadratic optimization problems.
10	`MSK_DPAR_DATA_TOL_CJ_LARGE` An element in c which is larger than this value in absolute terms causes a warning message to be printed.
26	`MSK_DPAR_INTPNT_NL_TOL_REL_STEP` Relative step size to the boundary for general nonlinear optimization problems.
36	`MSK_DPAR_INTPNT_TOL_STEP_SIZE` If the step size falls below the value of this parameter, then the interior-point optimizer assumes it is stalled. It it does not not make any progress.
32	`MSK_DPAR_INTPNT_TOL_PFEAS` Primal feasibility tolerance used for linear and quadratic optimization problems.
0	`MSK_DPAR_BASIS_REL_TOL_S` Maximum relative dual bound violation allowed in an optimal basic solution.
15	`MSK_DPAR_INTPNT_CO_TOL_INFEAS`

continued on next page

continued from previous page

	Controls when the conic interior-point optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.
47	`MSK_DPAR_MIO_TOL_ABS_RELAX_INT` Absolute relaxation tolerance of the integer constraints. I.e. $\min(\lvert x\rvert - \lfloor x \rfloor, \lceil x \rceil - \lvert x\rvert)$ is less than the tolerance then the integer restrictions assumed to be satisfied.
54	`MSK_DPAR_PRESOLVE_TOL_AIJ` Absolute zero tolerance employed for a_{ij} in the presolve.
42	`MSK_DPAR_MIO_MAX_TIME_APRX_OPT` Number of seconds spent by the mixed integer optimizer before the `MSK_DPAR_MIO_TOL_REL_RELAX_INT` is applied.
31	`MSK_DPAR_INTPNT_TOL_PATH` Controls how close the interior-point optimizer follows the central path. A large value of this parameter means the central is followed very closely. On numerical unstable problems it might worthwhile to increase this parameter.
20	`MSK_DPAR_INTPNT_NL_MERIT_BAL` Controls if the complementarity and infeasibility is converging to zero at about equal rates.
2	`MSK_DPAR_BASIS_TOL_X` Maximum absolute primal bound violation allowed in an optimal basic solution.
34	`MSK_DPAR_INTPNT_TOL_REL_GAP` Relative gap termination tolerance.
8	`MSK_DPAR_DATA_TOL_BOUND_WRN` If a bound value is larger than this value in absolute size, then a warning message is issued.
7	`MSK_DPAR_DATA_TOL_BOUND_INF` Any bound which in absolute value is greater than this parameter is considered infinite.
33	`MSK_DPAR_INTPNT_TOL_PSAFE` Controls the initial primal starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly and/or the constraint or variable bounds are very large, then it might be worthwhile to increase this value.
17	`MSK_DPAR_INTPNT_CO_TOL_NEAR_REL` If MOSEK cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.
4	`MSK_DPAR_CALLBACK_FREQ`

continued on next page

continued from previous page

	Controls the time between calls to the progress call-back function. Hence, if the value of this parameter is for example 10, then the call-back is called approximately each 10 seconds. A negative value is equivalent to infinity. In general frequent call-backs may hurt the performance.
24	`MSK_DPAR_INTPNT_NL_TOL_PFEAS` Primal feasibility tolerance used when a nonlinear model is solved.
21	`MSK_DPAR_INTPNT_NL_TOL_DFEAS` Dual feasibility tolerance used when a nonlinear model is solved.
48	`MSK_DPAR_MIO_TOL_REL_GAP` Relative optimality tolerance employed by the mixed integer optimizer.
27	`MSK_DPAR_INTPNT_TOL_DFEAS` Dual feasibility tolerance used for linear and quadratic optimization problems.
43	`MSK_DPAR_MIO_NEAR_TOL_ABS_GAP` Relaxed absolute optimality tolerance employed by the mixed integer optimizer. This termination criteria is delayed. See `MSK_DPAR_MIO_DISABLE_TERM_TIME` for details.
49	`MSK_DPAR_MIO_TOL_REL_RELAX_INT` Relative relaxation tolerance of the integer constraints. I.e. $\min(\lvert x\rvert - \lfloor x\rfloor, \lceil x\rceil - \lvert x\rvert)$ is less than the tolerance times $\lvert x\rvert$ then the integer restrictions assumed to be satisfied.
11	`MSK_DPAR_DATA_TOL_QIJ` Absolute zero tolerance for elements in Q matrices.
25	`MSK_DPAR_INTPNT_NL_TOL_REL_GAP` Relative gap termination tolerance for nonlinear problems.
18	`MSK_DPAR_INTPNT_CO_TOL_PFEAS` Primal feasibility tolerance used by the conic interior-point optimizer.

I.13 Double values

Value	Name Description
1e+30	`MSK_INFINITY` Definition of infinity.

I.14 Feasibility repair types

continued on next page

continued from previous page

Value	Name Description
0	`MSK_FEASREPAIR_OPTIMIZE_NONE` Do not optimize the feasibility repair problem.
2	`MSK_FEASREPAIR_OPTIMIZE_COMBINED` Minimize with original objective subject to minimal weighted violation of bounds.
1	`MSK_FEASREPAIR_OPTIMIZE_PENALTY` Minimize weighted sum of violations.

I.15 Integer information items.

Value	Name Description
10	`MSK_IINF_MIO_CONSTRUCT_SOLUTION` If this item has the value 0, then MOSEK did not try to construct an initial integer feasible solution. If the item has a positive value, then MOSEK successfully constructed an initial integer feasible solution.
62	`MSK_IINF_SIM_PRIMAL_INF_ITER` The number of iterations taken with primal infeasibility.
45	`MSK_IINF_RD_NUMCON` Number of constraints read.
71	`MSK_IINF_STO_NUM_A_CACHE_FLUSHES` Number of times the cache of A elements is flushed. A large number implies that `maxnumanz` is too small as well as an inefficient usage of MOSEK.
20	`MSK_IINF_MIO_NUMINT` Number of integer variables in the problem solved be the mixed integer optimizer.
67	`MSK_IINF_SOL_INT_PROSTA` Problem status of the integer solution. Updated after each optimization.
51	`MSK_IINF_RD_PROTYPE` Problem type.
73	`MSK_IINF_STO_NUM_A_TRANSPOSES` Number of times the A matrix is transposed. A large number implies that `maxnumanz` is too small or an inefficient usage of MOSEK. This will occur in particular if the code alternate between accessing rows and columns of A.
25	`MSK_IINF_MIO_TOTAL_NUM_CLIQUE_CUTS` Number of clique cuts.
68	`MSK_IINF_SOL_INT_SOLSTA`

continued on next page

continued from previous page

	Solution status of the integer solution. Updated after each optimization.
12	`MSK_IINF_MIO_NUM_ACTIVE_NODES` Number of active nodes in the branch and bound tree.
9	`MSK_IINF_INTPNT_SOLVE_DUAL` Non-zero if the interior-point optimizer is solving the dual problem.
39	`MSK_IINF_MIO_TOTAL_NUM_RELAX` Number of relaxations solved during the optimization.
28	`MSK_IINF_MIO_TOTAL_NUM_CUTS` Total number of cuts generated by the mixed integer optimizer.
2	`MSK_IINF_CACHE_SIZE_L2` L2 cache size used.
66	`MSK_IINF_SOL_BAS_SOLSTA` Solution status of the basic solution. Updated after each optimization.
50	`MSK_IINF_RD_NUMVAR` Number of variables read.
64	`MSK_IINF_SIM_SOLVE_DUAL` Is non-zero if dual problem is solved.
47	`MSK_IINF_RD_NUMINTVAR` Number of integer constrained variables read.
46	`MSK_IINF_RD_NUMCONE` Number of conic constraints read.
43	`MSK_IINF_OPTIMIZE_RESPONSE` The reponse code returned by optimize.
52	`MSK_IINF_SIM_DUAL_DEG_ITER` The number of dual degenerate iterations.
69	`MSK_IINF_SOL_ITR_PROSTA` Problem status of the interior-point solution. Updated after each optimization.
37	`MSK_IINF_MIO_TOTAL_NUM_OBJ_CUTS` Number of obj cuts.
11	`MSK_IINF_MIO_INITIAL_SOLUTION` Is non-zero if an initial integer solution is specified.
36	`MSK_IINF_MIO_TOTAL_NUM_LIFT_CUTS` Number of lift cuts.
8	`MSK_IINF_INTPNT_NUM_THREADS` Number of threads that the interior-point optimizer is using.
65	`MSK_IINF_SOL_BAS_PROSTA` Problem status of the basic solution. Updated after each optimization.
54	`MSK_IINF_SIM_DUAL_HOTSTART_LU` If 1 then a valid basis factorization of full rank was located and used by the dual simplex algorithm.

continued on next page

continued from previous page

70	`MSK_IINF_SOL_ITR_SOLSTA` Solution status of the interior-point solution. Updated after each optimization.
19	`MSK_IINF_MIO_NUMCON` Number of constraints in the problem solved be the mixed integer optimizer.
42	`MSK_IINF_OPT_NUMVAR` Number of variables in the problem solved when the optimizer is called
58	`MSK_IINF_SIM_NUMVAR` Number of variables in the problem solved by the simplex optimizer.
35	`MSK_IINF_MIO_TOTAL_NUM_LATTICE_CUTS` Number of lattice cuts.
63	`MSK_IINF_SIM_PRIMAL_ITER` Number of primal simplex iterations during the last optimization.
33	`MSK_IINF_MIO_TOTAL_NUM_GUB_COVER_CUTS` Number of GUB cover cuts.
44	`MSK_IINF_RD_NUMANZ` Number of non-zeros in A that is read.
5	`MSK_IINF_INTPNT_FACTOR_NUM_NZ` Number of non-zeros in factorization.
24	`MSK_IINF_MIO_TOTAL_NUM_CARDGUB_CUTS` Number of cardgub cuts.
16	`MSK_IINF_MIO_NUM_INTPNT_ITER` Number of interior-point iterations performed by the mixed-integer optimizer.
60	`MSK_IINF_SIM_PRIMAL_HOTSTART` If 1 then the primal simplex algorithm is solving from an advance basis.
27	`MSK_IINF_MIO_TOTAL_NUM_CONTRA_CUTS` Number of contra cuts.
0	`MSK_IINF_BI_ITER` Number of simplex pivots performed since invoking the basis identification procedure.
13	`MSK_IINF_MIO_NUM_BRANCH` Number of branches performed during the optimization.
3	`MSK_IINF_CONCURRENT_FASTEST_OPTIMIZER` The type of the optimizer that finished first in a concurrent optimization.
40	`MSK_IINF_MIO_USER_OBJ_CUT` If it is non-zero, then the objective cut is used.
32	`MSK_IINF_MIO_TOTAL_NUM_GOMORY_CUTS` Number of Gomory cuts.
1	`MSK_IINF_CACHE_SIZE_L1`

continued on next page

continued from previous page

	L1 cache size used.
29	MSK_IINF_MIO_TOTAL_NUM_DISAGG_CUTS Number of diasagg cuts.
48	MSK_IINF_RD_NUMQ Number of nonempty Q matrices read.
26	MSK_IINF_MIO_TOTAL_NUM_COEF_REDC_CUTS Number of coef. redc. cuts.
6	MSK_IINF_INTPNT_FACTOR_NUM_OFFCOL Number of columns in the constraint matrix (or Jacobian) that has an offending structure.
72	MSK_IINF_STO_NUM_A_REALLOC Number of times the storage for storing A has been changed. A large value may indicates that memory fragmentation may occur.
17	MSK_IINF_MIO_NUM_RELAX Number of relaxations solved during the optimization.
38	MSK_IINF_MIO_TOTAL_NUM_PLAN_LOC_CUTS Number of loc cuts.
23	MSK_IINF_MIO_TOTAL_NUM_BRANCH Number of branches performed during the optimization.
49	MSK_IINF_RD_NUMQNZ Number of Q non-zeros.
59	MSK_IINF_SIM_PRIMAL_DEG_ITER The number of primal degenerate iterations.
31	MSK_IINF_MIO_TOTAL_NUM_GCD_CUTS Number of gcd cuts.
53	MSK_IINF_SIM_DUAL_HOTSTART If 1 then the dual simplex algorithm is solving from an advance basis.
18	MSK_IINF_MIO_NUM_SIMPLEX_ITER Number of simplex iterations performed by the mixed-integer optimizer.
56	MSK_IINF_SIM_DUAL_ITER Number of dual simplex iterations during the last optimization.
55	MSK_IINF_SIM_DUAL_INF_ITER The number of iterations taken with dual infeasibility.
30	MSK_IINF_MIO_TOTAL_NUM_FLOW_COVER_CUTS Number of flow cover cuts.
21	MSK_IINF_MIO_NUMVAR Number of variables in the problem solved be the mixed integer optimizer.
15	MSK_IINF_MIO_NUM_INT_SOLUTIONS Number of integer feasible solutions that has been found.
61	MSK_IINF_SIM_PRIMAL_HOTSTART_LU If 1 then a valid basis factorization of full rank was located and used by the primal simplex algorithm.

continued on next page

continued from previous page

14	`MSK_IINF_MIO_NUM_CUTS` Number of cuts generated by the mixed integer optimizer.
4	`MSK_IINF_CPU_TYPE` The type of cpu detected.
7	`MSK_IINF_INTPNT_ITER` Number of interior-point iterations since invoking the interior-point optimizer.
34	`MSK_IINF_MIO_TOTAL_NUM_KNAPSUR_COVER_CUTS` Number of knapsack cover cuts.
41	`MSK_IINF_OPT_NUMCON` Number of constraints in the problem solved when the optimizer is called.
22	`MSK_IINF_MIO_TOTAL_NUM_BASIS_CUTS` Number of basis cuts.
57	`MSK_IINF_SIM_NUMCON` Number of constraints in the problem solved by the simplex optimizer.

I.16 Information item types

Value	Name Description
0	`MSK_INF_DOU_TYPE` Is a double information type.
1	`MSK_INF_INT_TYPE` Is an integer.

I.17 Input/output modes

Value	Name Description
0	`MSK_IOMODE_READ` The file is read-only.
1	`MSK_IOMODE_WRITE` The file is write-only. If the file exists then it is truncated when it is opened. Otherwise it is created when it is opened.
2	`MSK_IOMODE_READWRITE` The file is to read and written.

I.18 Integer parameters

Value	Name Description
157	`MSK_IPAR_SIM_STABILITY_PRIORITY` Controls how high priority the numerical stability should be given.
115	`MSK_IPAR_READ_ADD_CONE` Additional number of conic constraints that is made room for in the problem.
185	`MSK_IPAR_WRITE_MPS_STRICT` Controls whether the written MPS file satisfies the MPS format strictly or not.
21	`MSK_IPAR_INFEAS_REPORT_AUTO` Controls whether an infeasibility report is automatically produced after the optimization if the problem is primal or dual infeasible.
87	`MSK_IPAR_MIO_NODE_OPTIMIZER` Controls which optimizer is employed at the non-root nodes in the mixed integer optimizer.
109	`MSK_IPAR_PRESOLVE_LEVEL` Currently not used.
117	`MSK_IPAR_READ_ADD_VAR` Additional number of variables that is made room for in the problem.
112	`MSK_IPAR_PRESOLVE_USE` Controls whether the presolve is applied to a problem before it is optimized.
64	`MSK_IPAR_LOG_SENSITIVITY_OPT` Controls the amount of logging from the optimizers employed during the sensitivity analysis. 0 means no logging information is produced.
101	`MSK_IPAR_OPF_WRITE_SOL_ITG` If `MSK_IPAR_OPF_WRITE_SOLUTIONS` is `MSK_ON` and an integer solution is defined, write the integer solution in OPF files.
167	`MSK_IPAR_WRITE_BAS_HEAD` Controls whether the header section is written to the basic solution file.
74	`MSK_IPAR_MIO_BRANCH_PRIORITIES_USE` Controls whether branching priorities are used by the mixed integer optimizer.
78	`MSK_IPAR_MIO_CUT_LEVEL_TREE` Controls the cut level employed by the mixed integer optimizer at the tree. See `MSK_IPAR_MIO_CUT_LEVEL_ROOT` for an explanation of the parameter values.
169	`MSK_IPAR_WRITE_DATA_COMPRESSED` Controls whether the data file is compressed while it is written. 0 means no compression while higher values mean more compression.
130	`MSK_IPAR_READ_MPS_RELAX` If this option is turned on, then the relaxation of the MIP will be read.

continued on next page

continued from previous page

98	`MSK_IPAR_OPF_WRITE_PARAMETERS` Write a parameter section in an OPF file.
119	`MSK_IPAR_READ_CON` Expected maximum number of constraints to be read. The option is only used by fast MPS and LP file readers.
177	`MSK_IPAR_WRITE_INT_VARIABLES` Controls whether the variables section is written to the integer solution file.
113	`MSK_IPAR_READ_ADD_ANZ` Additional number of non-zeros in A that is made room for in the problem.
32	`MSK_IPAR_INTPNT_ORDER_METHOD` Controls the ordering strategy used by the interior-point optimizer when factorizing the Newton equation system.
102	`MSK_IPAR_OPF_WRITE_SOL_ITR` If `MSK_IPAR_OPF_WRITE_SOLUTIONS` is `MSK_ON` and an interior solution is defined, write the interior solution in OPF files.
63	`MSK_IPAR_LOG_SENSITIVITY` Controls the amount of logging during the sensitivity analysis. 0: Means no logging information is produced. 1: Timing information is printed. 2: Sensitivity results are printed.
133	`MSK_IPAR_READ_QNZ` Expected maximum number of Q non-zeros to be read. The option is used only by MPS and LP file readers.
53	`MSK_IPAR_LOG_INFEAS_ANA` Controls amount of output printed by the infeasibility analyzer procedures. A higher level implies that more information is logged.
152	`MSK_IPAR_SIM_PRIMAL_SELECTION` Controls the choice of the incoming variable, known as the selection strategy, in the primal simplex optimizer.
175	`MSK_IPAR_WRITE_INT_CONSTRAINTS` Controls whether the constraint section is written to the integer solution file.
180	`MSK_IPAR_WRITE_LP_STRICT_FORMAT` Controls whether LP output files satisfy the LP format strictly.
138	`MSK_IPAR_SENSITIVITY_TYPE` Controls which type of sensitivity analysis is to be performed.
141	`MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION`

continued on next page

continued from previous page

	The dual simplex optimizer can use a so-called restricted selection/pricing strategy to chooses the outgoing variable. Hence, if restricted selection is applied, then the dual simplex optimizer first choose a subset of all the potential outgoing variables. Next, for some time it will choose the outgoing variable only among the subset. From time to time the subset is redefined. A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.
56	`MSK_IPAR_LOG_MIO_FREQ` Controls how frequent the mixed integer optimizer prints the log line. It will print line every time `MSK_IPAR_LOG_MIO_FREQ` relaxations have been solved.
100	`MSK_IPAR_OPF_WRITE_SOL_BAS` If `MSK_IPAR_OPF_WRITE_SOLUTIONS` is `MSK_ON` and a basic solution is defined, include the basic solution in OPF files.
91	`MSK_IPAR_MIO_ROOT_OPTIMIZER` Controls which optimizer is employed at the root node in the mixed integer optimizer.
172	`MSK_IPAR_WRITE_FREE_CON` Controls whether the free constraints are written to the data file.
107	`MSK_IPAR_PRESOLVE_ELIM_FILL` Controls the maximum amount of fill-in that can be created during the elimination phase of the presolve. This parameter times (`numcon`+`numvar`) denotes the amount of fill-in.
93	`MSK_IPAR_NONCONVEX_MAX_ITERATIONS` Maximum number of iterations that can be used by the nonconvex optimizer.
82	`MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER`
173	`MSK_IPAR_WRITE_GENERIC_NAMES` Controls whether the generic names or user-defined names are used in the data file.
165	`MSK_IPAR_WARNING_LEVEL` Warning level.
46	`MSK_IPAR_LOG_BI_FREQ` Controls how frequent the optimizer outputs information about the basis identification and how frequent the user-defined call-back function is called.
61	`MSK_IPAR_LOG_PRESOLVE` Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.
8	`MSK_IPAR_CHECK_CTRL_C`

continued on next page

continued from previous page

	Specifies whether MOSEK should check for `<ctrl>+<c>` key presses. In case it has, then control is returned to the user program. In case a user-defined ctrl-c function is defined then that is used to check for ctrl-c. Otherwise the system procedure `signal` is used.
116	`MSK_IPAR_READ_ADD_QNZ` Additional number of non-zeros in the Q matrices that is made room for in the problem.
187	`MSK_IPAR_WRITE_SOL_CONSTRAINTS` Controls whether the constraint section is written to the solution file.
31	`MSK_IPAR_INTPNT_OFF_COL_TRH` Controls how many offending columns are detected in the Jacobian of the constraint matrix. 1 means aggressive detection, higher values mean less aggressive detection. 0 means no detection.
118	`MSK_IPAR_READ_ANZ` Expected maximum number of A non-zeros to be read. The option is used only by fast MPS and LP file readers.
86	`MSK_IPAR_MIO_MODE` Controls whether the optimizer includes the integer restrictions when solving a (mixed) integer optimization problem.
124	`MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU` If this option is turned on, MOSEK will drop variables that are defined for the first time in the bounds section.
65	`MSK_IPAR_LOG_SIM` Controls amount of output printed by the simplex optimizer. A higher level implies that more information is logged.
58	`MSK_IPAR_LOG_OPTIMIZER` Controls the amount of general optimizer information that is logged.
164	`MSK_IPAR_SOLUTION_CALLBACK` Indicates whether solution call-backs will be performed during the optimization.
66	`MSK_IPAR_LOG_SIM_FREQ` Controls how frequent the simplex optimizer outputs information about the optimization and how frequent the user-defined call-back function is called.
57	`MSK_IPAR_LOG_NONCONVEX` Controls amount of output printed by the nonconvex optimizer.
17	`MSK_IPAR_FEASREPAIR_OPTIMIZE` Controls which type of feasibility analysis is to be performed.
179	`MSK_IPAR_WRITE_LP_QUOTED_NAMES` If this option is turned on, then MOSEK will quote invalid LP names when writing an LP file.
49	`MSK_IPAR_LOG_FACTOR`

continued on next page

continued from previous page

	If turned on, then the factor log lines are added to the log.
125	`MSK_IPAR_READ_LP_QUOTED_NAMES`
	If a name is in quotes when reading an LP file, the quotes will be removed.
184	`MSK_IPAR_WRITE_MPS_QUOTED_NAMES`
	If a name contains spaces (blanks) when writing an MPS file, then the quotes will be removed.
131	`MSK_IPAR_READ_MPS_WIDTH`
	Controls the maximal number of chars allowed in one line of the MPS file.
59	`MSK_IPAR_LOG_ORDER`
	If turned on, then factor lines are added to the log.
77	`MSK_IPAR_MIO_CUT_LEVEL_ROOT`
	Controls the cut level employed by the mixed integer optimizer at the root node. A negative value means a default value determined by the mixed integer optimizer is used. By adding the appropriate values from the following table the employed cut types can be controlled.
	GUB cover +2 Flow cover +4 Lifting +8 Plant location +16 Disaggregation +32 Knapsack cover +64 Lattice +128 Gomory +256 Coefficient reduction +512 GCD +1024 Obj. integrality +2048
132	`MSK_IPAR_READ_Q_MODE`
	Controls how the Q matrices are read from the MPS file.
88	`MSK_IPAR_MIO_NODE_SELECTION`
	Controls the node selection strategy employed by the mixed integer optimizer.
163	`MSK_IPAR_SOL_READ_WIDTH`
	Controls the maximal acceptable width of line in the solutions when read by MOSEK.
72	`MSK_IPAR_MAXNUMANZ_DOUBLE_TRH`
	Whenever MOSEK runs out of storage for the A matrix, it will double the value for `maxnumanz` until `maxnumnza` reaches the value of this parameter. When this threshold is reached it will use a slower increase.
29	`MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS`

continued on next page

continued from previous page

	Maximum number of steps to be used by the iterative refinement of the search direction. A negative value implies that the optimizer Chooses the maximum number of iterative refinement steps.
114	`MSK_IPAR_READ_ADD_CON` Additional number of constraints that is made room for in the problem.
47	`MSK_IPAR_LOG_CONCURRENT` Controls amount of output printed by the concurrent optimizer.
67	`MSK_IPAR_LOG_SIM_MINOR` Currently not in use.
144	`MSK_IPAR_SIM_MAX_ITERATIONS` Maximum number of iterations that can be used by a simplex optimizer.
27	`MSK_IPAR_INTPNT_MAX_ITERATIONS` Controls the maximum number of iterations allowed in the interior-point optimizer.
15	`MSK_IPAR_CPU_TYPE` Specifies the CPU type. By default MOSEK tries to auto detect the CPU type. Therefore, we recommend to change this parameter only if the auto detection does not work properly.
45	`MSK_IPAR_LOG_BI` Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.
28	`MSK_IPAR_INTPNT_MAX_NUM_COR` Controls the maximum number of correctors allowed by the multiple corrector procedure. A negative value means that MOSEK is making the choice.
178	`MSK_IPAR_WRITE_LP_LINE_WIDTH` Maximum width of line in an LP file written by MOSEK.
162	`MSK_IPAR_SOL_READ_NAME_WIDTH` When a solution is read by MOSEK and some constraint, variable or cone names contain blanks, then a maximum name width much be specified. A negative value implies that no name contain blanks.
40	`MSK_IPAR_LICENSE_DEBUG` This option is used to turn on debugging of the incense manager.
43	`MSK_IPAR_LICENSE_WAIT` If all licenses are in use MOSEK returns with an error code. However, by turning on this parameter MOSEK will wait for an available license.
174	`MSK_IPAR_WRITE_GENERIC_NAMES_IO` Index origin used in generic names.
10	`MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS` The maximum number of simultaneous optimizations that will be started by the concurrent optimizer.

continued on next page

continued from previous page

153	`MSK_IPAR_SIM_REFACTOR_FREQ` Controls how frequent the basis is refactorized. The value 0 means that the optimizer determines the best point of refactorization. It is strongly recommended NOT to change this parameter.
142	`MSK_IPAR_SIM_DUAL_SELECTION` Controls the choice of the incoming variable, known as the selection strategy, in the dual simplex optimizer.
156	`MSK_IPAR_SIM_SOLVE_FORM` Controls whether the primal or the dual problem is solved by the primal-/dual- simplex optimizer.
7	`MSK_IPAR_CHECK_CONVEXITY` Specify the level of convexity check on quadratic problems
70	`MSK_IPAR_LP_WRITE_IGNORE_INCOMPATIBLE_ITEMS` Controls the result of writing a problem containing incompatible items to an LP file.
134	`MSK_IPAR_READ_TASK_IGNORE_PARAM` Controls whether MOSEK should ignore the parameter setting defined in the task file and use the default parameter setting instead.
9	`MSK_IPAR_CHECK_TASK_DATA` If this feature is turned on, then the task data is checked for bad values i.e. NaNs. before an optimization is performed.
54	`MSK_IPAR_LOG_INTPNT` Controls amount of output printed printed by the interior-point optimizer. A higher level implies that more information is logged.
55	`MSK_IPAR_LOG_MIO` Controls the log level for the mixed integer optimizer. A higher level implies that more information is logged.
143	`MSK_IPAR_SIM_HOTSTART` Controls the type of hot-start that the simplex optimizer perform.
60	`MSK_IPAR_LOG_PARAM` Controls the amount of information printed out about parameter changes.
170	`MSK_IPAR_WRITE_DATA_FORMAT` Controls the file format when writing task data to a file.
73	`MSK_IPAR_MIO_BRANCH_DIR` Controls whether the mixed integer optimizer is branching up or down by default.
25	`MSK_IPAR_INTPNT_FACTOR_DEBUG_LVL` Controls factorization debug level.
18	`MSK_IPAR_FLUSH_STREAM_FREQ` Controls how frequent the message and log streams are flushed. A value of 0 means that it is never flushed. Otherwise a larger value results in less frequent flushes.

continued on next page

continued from previous page

161	`MSK_IPAR_SOL_QUOTED_NAMES` If this options is turned on, then MOSEK will quote names that contains blanks while writing the solution file. Moreover when reading leading and trailing quotes will be stripped of.
41	`MSK_IPAR_LICENSE_PAUSE_TIME` If `MSK_IPAR_LICENSE_WAIT`=`MSK_ON` and no license is available, then MOSEK sleeps a number of micro seconds between each check of whether a license as become free.
89	`MSK_IPAR_MIO_PRESOLVE_AGGREGATE` Controls whether the presolve used by the mixed integer optimizer tries to aggregate the constraints.
190	`MSK_IPAR_WRITE_TASK_INC_SOL` Controls whether the solutions are stored in the task file too.
38	`MSK_IPAR_LICENSE_CACHE_TIME` Controls the amount of time a license is cached in the MOSEK environment for reuse. Checking out a license from the license server has a small overhead. Therefore, if a large number of optimizations is performed within a small amount of time, it is efficient to cache the license in the MOSEK environment for later use. This way a number of license check outs from the license server is avoided. If a license has not been used in the given amount of time,MOSEK will automatically check in the license. To disable license caching set the value to 0.
4	`MSK_IPAR_BI_MAX_ITERATIONS` Controls the maximum number of simplex iterations allowed to optimize a basis after the basis identification.
103	`MSK_IPAR_OPF_WRITE_SOLUTIONS` Enable inclusion of solutions in the OPF files.

147	`MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART`
	This parameter controls has large the network component in “relative” terms has to be before it is exploited in a simplex hot-start. The network component should be equal or larger than
	`max(MSK_IPAR_SIM_NETWORK_DETECT,MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART)`
	before it is exploited. If this value is larger than 100 the network flow component is never detected or exploited.

110	`MSK_IPAR_PRESOLVE_LINDEP_USE` Controls whether the linear constraints are checked for linear dependencies.
106	`MSK_IPAR_PARAM_READ_IGN_ERROR` If turned on, then errors in paramter settings is ignored.
96	`MSK_IPAR_OPF_WRITE_HEADER` Write a text header with date and MOSEK version in an OPF file.
76	`MSK_IPAR_MIO_CONT_SOL`

continued on next page

continued from previous page

	Controls the meaning of the interior-point and basic solutions in MIP problems.
94	`MSK_IPAR_OBJECTIVE_SENSE` If the objective sense for the task is undefined, then the value of this parameter is used as the default objective sense.
176	`MSK_IPAR_WRITE_INT_HEAD` Controls whether the header section is written to the integer solution file.
36	`MSK_IPAR_INTPNT_STARTING_POINT` Starting point used by the interior-point optimizer.
44	`MSK_IPAR_LOG` Controls the amount of log information. The value 0 implies that all log information is suppressed. A higher level implies that more information is logged. Please note that if a task is employed to solve a sequence of optimization problems the value of this parameter is reduced by the value of `MSK_IPAR_LOG_CUT_SECOND_OPT` for the second and any subsequent optimizations.
14	`MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX` Priority of the primal simplex algorithm when selecting solvers for concurrent optimization.
128	`MSK_IPAR_READ_MPS_OBJ_SENSE` If turned on, the MPS reader uses the objective sense section. Otherwise the MPS reader ignores it.
68	`MSK_IPAR_LOG_SIM_NETWORK_FREQ` Controls how frequent the network simplex optimizer outputs information about the optimization and how frequent the user-defined call-back function is called. The network optimizer will use a logging frequency equal to `MSK_IPAR_LOG_SIM_FREQ` times `MSK_IPAR_LOG_SIM_NETWORK_FREQ`.
24	`MSK_IPAR_INTPNT_DIFF_STEP` Controls whether different step sizes are allowed in the primal and dual space.
155	`MSK_IPAR_SIM_SCALING` Controls how the problem is scaled before a simplex optimizer is used.
181	`MSK_IPAR_WRITE_LP_TERMS_PER_LINE` Maximum number of terms on a single line in an LP file written by MOSEK. 0 means unlimited.
136	`MSK_IPAR_SENSITIVITY_ALL` Not applicable.
160	`MSK_IPAR_SOL_FILTER_KEEP_RANGED` If turned on, then ranged constraints and variables are written to the solution file independent of the filter setting.

continued on next page

continued from previous page

2	`MSK_IPAR_BI_IGNORE_MAX_ITER` If the parameter `MSK_IPAR_INTPNT_BASIS` has the value `MSK_BI_NO_ERROR` and the interior-point optimizer has terminated due to maximum number of iterations, then basis identification is performed if this parameter has the value `MSK_ON`.
50	`MSK_IPAR_LOG_FEASREPAIR` Controls the amount of output printed when performing feasibility repair.
35	`MSK_IPAR_INTPNT_SOLVE_FORM` Controls whether the primal or the dual problem is solved.
95	`MSK_IPAR_OPF_MAX_TERMS_PER_LINE` The maximum number of terms (linear and quadratic) per line when an OPF file is written.
186	`MSK_IPAR_WRITE_PRECISION` Controls the precision with which `double` numbers are printed in the MPS data file. In general it is not worthwhile to use a value higher than 15.
191	`MSK_IPAR_WRITE_XML_MODE` Controls if linear coefficients should be written by row or column when writing in the XML file format.
33	`MSK_IPAR_INTPNT_REGULARIZATION_USE` Controls whether regularization is allowed.
1	`MSK_IPAR_BI_CLEAN_OPTIMIZER` Controls which simplex optimizer is used in the clean-up phase.
192	`MSK_IPAR_MIO_PRESOLVE_PROBING` Controls whether the mixed integer presolve performs probing. Probing can be very time consuming.
37	`MSK_IPAR_LICENSE_ALLOW_OVERUSE` Controls if license overuse is allowed when caching licenses
20	`MSK_IPAR_INFEAS_PREFER_PRIMAL` If both certificates of primal and dual infeasibility are supplied then only the primal is used when this option is turned on.
168	`MSK_IPAR_WRITE_BAS_VARIABLES` Controls whether the variables section is written to the basic solution file.
69	`MSK_IPAR_LOG_STORAGE` When turned on, MOSEK prints messages regarding the storage usage and allocation.
90	`MSK_IPAR_MIO_PRESOLVE_USE` Controls whether presolve is performed by the mixed integer optimizer.
111	`MSK_IPAR_PRESOLVE_LINDEP_WORK_LIM`

continued on next page

continued from previous page

	Is used to limit the amount of work that can done to locate linear dependencies. In general the higher value this parameter is given the less work can be used. However, a value of 0 means no limit on the amount work that can be used.
23	MSK_IPAR_INTPNT_BASIS Controls whether the interior-point optimizer also computes an optimal basis.
48	MSK_IPAR_LOG_CUT_SECOND_OPT If a task is employed to solve a sequence of optimization problems, then the value of the log levels is reduced by the value of this parameter. E.g MSK_IPAR_LOG and MSK_IPAR_LOG_SIM are reduced by the value of this parameter for the second and any subsequent optimizations.
127	MSK_IPAR_READ_MPS_KEEP_INT Controls whether MOSEK should keep the integer restrictions on the variables while reading the MPS file.
85	MSK_IPAR_MIO_MAX_NUM_SOLUTIONS The mixed integer optimizer can be terminated after a certain number of different feasible solutions has been located. If this parameter has the value n and n is strictly positive, then the mixed integer optimizer will be terminated when n feasible solutions have been located.
39	MSK_IPAR_LICENSE_CHECK_TIME The parameter specifies the number of seconds between the checks of all the active licenses in the MOSEK environment license cache. These checks are performed to determine if the licenses should be returned to the server.
189	MSK_IPAR_WRITE_SOL_VARIABLES Controls whether the variables section is written to the solution file.
137	MSK_IPAR_SENSITIVITY_OPTIMIZER Controls which optimizer is used for optimal partition sensitivity analysis.
182	MSK_IPAR_WRITE_MPS_INT Controls if marker records are written to the MPS file to indicate whether variables are integer restricted.
145	MSK_IPAR_SIM_MAX_NUM_SETBACKS Controls how many setbacks are allowed within a simplex optimizer. A setback is an event where the optimizer moves in the wrong direction. This is impossible in theory but may happen due to numerical problems.
16	MSK_IPAR_DATA_CHECK If this option is turned on, then extensive data checking is enabled. It will slow down MOSEK but on the other hand help locating bugs.
12	MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX

continued on next page

continued from previous page

	Priority of the free simplex optimizer when selecting solvers for concurrent optimization.
123	`MSK_IPAR_READ_KEEP_FREE_CON` Controls whether the free constraints are included in the problem.
51	`MSK_IPAR_LOG_FILE` If turned on, then some log info is printed when a file is written or read.
13	`MSK_IPAR_CONCURRENT_PRIORITY_INTPNT` Priority of the interior-point algorithm when selecting solvers for concurrent optimization.
149	`MSK_IPAR_SIM_NON_SINGULAR` Controls if the simplex optimizer ensures a non-singular basis, if possible.
171	`MSK_IPAR_WRITE_DATA_PARAM` If this option is turned on the parameter settings are written to the data file as parameters.
139	`MSK_IPAR_SIM_DEGEN` Controls how aggressive degeneration is approached.
97	`MSK_IPAR_OPF_WRITE_HINTS` Write a hint section with problem dimensions in the beginning of an OPF file.
108	`MSK_IPAR_PRESOLVE_ELIMINATOR_USE` Controls whether free or implied free variables are eliminated from the problem.
0	`MSK_IPAR_ALLOC_ADD_QNZ` Additional number of Q non-zeros that are allocated space for when `numanz` exceeds `maxnumqnz` during addition of new Q entries.
126	`MSK_IPAR_READ_MPS_FORMAT` Controls how strictly the MPS file reader interprets the MPS format.
105	`MSK_IPAR_PARAM_READ_CASE_NAME` If turned on, then names in the parameter file are case sensitive.
129	`MSK_IPAR_READ_MPS_QUOTED_NAMES` If a name is in quotes when reading an MPS file, then the quotes will be removed.
11	`MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX` Priority of the dual simplex algorithm when selecting solvers for concurrent optimization.
183	`MSK_IPAR_WRITE_MPS_OBJ_SENSE` If turned off, the objective sense section is not written to the MPS file.
30	`MSK_IPAR_INTPNT_NUM_THREADS` Controls the number of threads employed by the interior-point optimizer.
83	`MSK_IPAR_MIO_MAX_NUM_BRANCHES`

continued on next page

continued from previous page

	Maximum number of branches allowed during the branch and bound search. A negative value means infinite.
150	`MSK_IPAR_SIM_PRIMAL_CRASH` Controls whether crashing is performed in the primal simplex optimizer. In general, if a basis consists of more than (100-this parameter value)% fixed variables, then a crash will be performed.
75	`MSK_IPAR_MIO_CONSTRUCT_SOL` If set to `MSK_ON` and all integer variables have been given a value for which a feasible MIP solution exists, then MOSEK generates an initial solution to the MIP by fixing all integer values and solving for the continuous variables.
92	`MSK_IPAR_MIO_STRONG_BRANCH` The value specifies the depth from the root in which strong branching is used. A negative value means that the optimizer chooses a default value automatically.
151	`MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION` The primal simplex optimizer can use a so-called restricted selection/pricing strategy to chooses the outgoing variable. Hence, if restricted selection is applied, then the primal simplex optimizer first choose a subset of all the potential incoming variables. Next, for some time it will choose the incoming variable only among the subset. From time to time the subset is redefined. A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.
120	`MSK_IPAR_READ_CONE` Expected maximum number of conic constraints to be read. The option is used only by fast MPS and LP file readers.
104	`MSK_IPAR_OPTIMIZER` Controls which optimizer is used to optimize the task.
71	`MSK_IPAR_MAX_NUM_WARNINGS` Waning level. A higher value results in more warnings.
42	`MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS` Controls whether license features expire warnings are suppressed.
188	`MSK_IPAR_WRITE_SOL_HEAD` Controls whether the header section is written to the solution file.
166	`MSK_IPAR_WRITE_BAS_CONSTRAINTS` Controls whether the constraint section is written to the basic solution file.
79	`MSK_IPAR_MIO_FEASPUMP_LEVEL`

continued on next page

continued from previous page

	Feasibility pump is a heuristic designed to compute an initial feasible solution. A value of 0 implies that the feasibility pump heuristic is not used. A value of -1 implies that the mixed integer optimizer decides how the feasibility pump heuristic is used. A larger value than 1 implies that the feasibility pump is employed more aggressively. Normally a value beyond 3 is not worthwhile.
19	`MSK_IPAR_INFEAS_GENERIC_NAMES`
	Controls whether generic names are used when an infeasible subproblem is created.
146	`MSK_IPAR_SIM_NETWORK_DETECT`
	The simplex optimizer is capable of exploiting a network flow component in a problem. However it is only worthwhile to exploit the network flow component if it is sufficiently large. This parameter controls how large the network component has to be in “relative” terms before it is exploited. For instance a value of 20 means at least 20% of the model should be a network before it is exploited. If this value is larger than 100 the network flow component is never detected or exploited.
62	`MSK_IPAR_LOG_RESPONSE`
	Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.
22	`MSK_IPAR_INFEAS_REPORT_LEVEL`
	Controls the amount of information presented in an infeasibility report. Higher values imply more information.
5	`MSK_IPAR_CACHE_SIZE_L1`
	Specifies the size of the cache of the computer. This parameter is potentially very important for the efficiency on computers if MOSEK cannot determine the cache size automatically. If the cache size is negative, then MOSEK tries to determine the value automatically.
6	`MSK_IPAR_CACHE_SIZE_L2`
	Specifies the size of the cache of the computer. This parameter is potentially very important for the efficiency on computers where MOSEK cannot determine the cache size automatically. If the cache size is negative, then MOSEK tries to determine the value automatically.
158	`MSK_IPAR_SIM_SWITCH_OPTIMIZER`
	The simplex optimizer sometimes chooses to solve the dual problem instead of the primal problem. This implies that if you have chosen to use the dual simplex optimizer and the problem is dualized, then it actually makes sense to use the primal simplex optimizer instead. If this parameter is on and the problem is dualized and furthermore the simplex optimizer is chosen to be the primal (dual) one, then it is switched to the dual (primal).
121	`MSK_IPAR_READ_DATA_COMPRESSED`

continued on next page

continued from previous page

	If this option is turned on,it is assumed that the data file is compressed.
3	`MSK_IPAR_BI_IGNORE_NUM_ERROR` If the parameter `MSK_IPAR_INTPNT_BASIS` has the value `MSK_BI_NO_ERROR` and the interior-point optimizer has terminated due to a numerical problem, then basis identification is performed if this parameter has the value `MSK_ON`.
99	`MSK_IPAR_OPF_WRITE_PROBLEM` Write objective, constraints, bounds etc. to an OPF file.
122	`MSK_IPAR_READ_DATA_FORMAT` Format of the data file to be read.
140	`MSK_IPAR_SIM_DUAL_CRASH` Controls whether crashing is performed in the dual simplex optimizer. In general if a basis consists of more than (100-this parameter value)% fixed variables, then a crash will be performed.
148	`MSK_IPAR_SIM_NETWORK_DETECT_METHOD` Controls which type of detection method the network extraction should use.
135	`MSK_IPAR_READ_VAR` Expected maximum number of variable to be read. The option is used only by MPS and LP file readers.
52	`MSK_IPAR_LOG_HEAD` If turned on, then a header line is added to the log.
154	`MSK_IPAR_SIM_SAVE_LU` Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.
26	`MSK_IPAR_INTPNT_FACTOR_METHOD` Controls the method used to factor the Newton equation system.
84	`MSK_IPAR_MIO_MAX_NUM_RELAXS` Maximum number of relaxations allowed during the branch and bound search. A negative value means infinite.
159	`MSK_IPAR_SOL_FILTER_KEEP_BASIC` If turned on, then basic and super basic constraints and variables are written to the solution file independent of the filter setting.
80	`MSK_IPAR_MIO_HEURISTIC_LEVEL` Controls the heuristic employed by the mixed integer optimizer to locate an initial good integer feasible solution. A value of zero means the heuristic is not used at all. A larger value than 0 means that a gradually more sophisticated heuristic is used which is computationally more expensive. A negative value implies that the optimizer chooses the heuristic. Normally a value around 3 to 5 should be optimal.
81	`MSK_IPAR_MIO_KEEP_BASIS`

continued on next page

continued from previous page

	Controls whether the integer presolve keeps bases in memory. This speeds on the solution process at cost of bigger memory consumption.
34	`MSK_IPAR_INTPNT_SCALING` Controls how the problem is scaled before the interior-point optimizer is used.

I.19 Bound keys

Value	Name Description
0	`MSK_MARK_LO` The lower bound is selected for sensitivity analysis.
1	`MSK_MARK_UP` The upper bound is selected for sensitivity analysis.

I.20 Continuous mixed integer solution type

Value	Name Description
2	`MSK_MIO_CONT_SOL_ITG` The reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. A solution is only reported in case the problem has a primal feasible solution.
0	`MSK_MIO_CONT_SOL_NONE` No interior-point or basic solution are reported when the mixed integer optimizer is used.
1	`MSK_MIO_CONT_SOL_ROOT` The reported interior-point and basic solutions are a solution to the root node problem when mixed integer optimizer is used.
3	`MSK_MIO_CONT_SOL_ITG_REL` In case the problem is primal feasible then the reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. If the problem is primal infeasible, then the solution to the root node problem is reported.

I.21 Integer restrictions

Value	Name Description
0	`MSK_MIO_MODE_IGNORED` The integer constraints are ignored and the problem is solved as a continuous problem.
2	`MSK_MIO_MODE_LAZY` Integer restrictions should be satisfied if an optimizer is available for the problem.
1	`MSK_MIO_MODE_SATISFIED` Integer restrictions should be satisfied.

I.22 Mixed integer node selection types

Value	Name Description
5	`MSK_MIO_NODE_SELECTION_PSEUDO` The optimizer employs selects the node based on a pseudo cost estimate.
4	`MSK_MIO_NODE_SELECTION_HYBRID` The optimizer employs a hybrid strategy.
0	`MSK_MIO_NODE_SELECTION_FREE` The optimizer decides the node selection strategy.
3	`MSK_MIO_NODE_SELECTION_WORST` The optimizer employs a worst bound node selection strategy.
2	`MSK_MIO_NODE_SELECTION_BEST` The optimizer employs a best bound node selection strategy.
1	`MSK_MIO_NODE_SELECTION_FIRST` The optimizer employs a depth first node selection strategy.

I.23 MPS file format type

Value	Name Description
0	`MSK_MPS_FORMAT_STRICT` It is assumed that the input file satisfies the MPS format strictly.
1	`MSK_MPS_FORMAT_RELAXED` It is assumed that the input file satisfies a slightly relaxed version of the MPS format.
2	`MSK_MPS_FORMAT_FREE` It is assumed that the input file satisfies the free MPS format. This implies that spaces are not allowed in names. Otherwise the format is free.

I.24 Message keys

Value	Name Description
1000	`MSK_MSG_READING_FILE` None
1001	`MSK_MSG_WRITING_FILE` None
1100	`MSK_MSG_MPS_SELECTED` None

I.25 Network detection method

Value	Name Description
1	`MSK_NETWORK_DETECT_SIMPLE` The network detection should use a very simple heuristic.
2	`MSK_NETWORK_DETECT_ADVANCED` The network detection should use a more advanced heuristic.
0	`MSK_NETWORK_DETECT_FREE` The network detection is free.

I.26 Objective sense types

Value	Name Description
1	`MSK_OBJECTIVE_SENSE_MINIMIZE` The problem should be minimized.
0	`MSK_OBJECTIVE_SENSE_UNDEFINED` The objective sense is undefined.
2	`MSK_OBJECTIVE_SENSE_MAXIMIZE` The problem should be maximized.

I.27 On/off

Value	Name Description
1	`MSK_ON` Switch the option on.

continued on next page

continued from previous page

0	`MSK_OFF` Switch the option off.

I.28 Optimizer types

Value	Name Description
1	`MSK_OPTIMIZER_INTPNT` The interior-point optimizer is used.
9	`MSK_OPTIMIZER_CONCURRENT` The optimizer for nonconvex nonlinear problems.
7	`MSK_OPTIMIZER_MIXED_INT` The mixed integer optimizer.
5	`MSK_OPTIMIZER_DUAL_SIMPLEX` The dual simplex optimizer is used.
0	`MSK_OPTIMIZER_FREE` The optimizer is chosen automatically.
2	`MSK_OPTIMIZER_CONIC` Another cone optimizer.
8	`MSK_OPTIMIZER_NONCONVEX` The optimizer for nonconvex nonlinear problems.
3	`MSK_OPTIMIZER_QCONE` The Qcone optimizer is used.
4	`MSK_OPTIMIZER_PRIMAL_SIMPLEX` The primal simplex optimizer is used.
6	`MSK_OPTIMIZER_FREE_SIMPLEX` Either the primal or the dual simplex optimizer is used.

I.29 Ordering strategies

Value	Name Description
5	`MSK_ORDER_METHOD_NONE` No ordering is used.
2	`MSK_ORDER_METHOD_APPMINLOC2` A variant of the approximate minimum local-fill-in ordering is used.
1	`MSK_ORDER_METHOD_APPMINLOC1` Approximate minimum local-fill-in ordering is used.
4	`MSK_ORDER_METHOD_GRAPHPAR2` An alternative graph partitioning based ordering.
0	`MSK_ORDER_METHOD_FREE`

continued on next page

continued from previous page

	The ordering method is chosen automatically.
3	`MSK_ORDER_METHOD_GRAPHPAR1` Graph partitioning based ordering.

I.30 Parameter type

Value	Name Description
0	`MSK_PAR_INVALID_TYPE` Not a valid parameter.
3	`MSK_PAR_STR_TYPE` Is a string parameter.
1	`MSK_PAR_DOU_TYPE` Is a double parameter.
2	`MSK_PAR_INT_TYPE` Is an integer parameter.

I.31 Presolve method.

Value	Name Description
1	`MSK_PRESOLVE_MODE_ON` The problem is presolved before it is optimized.
0	`MSK_PRESOLVE_MODE_OFF` The problem is not presolved before it is optimized.
2	`MSK_PRESOLVE_MODE_FREE` It is decided automatically whether to presolve before the problem is optimized.

I.32 Problem data items

Value	Name Description
0	`MSK_PI_VAR` Item is a variable.
2	`MSK_PI_CONE` Item is a cone.
1	`MSK_PI_CON` Item is a constraint.

I.33 Problem types

Value	Name Description
2	`MSK_PROBTYPE_QCQO` The problem is a quadratically constrained optimization problem.
0	`MSK_PROBTYPE_LO` The problem is a linear optimization problem.
4	`MSK_PROBTYPE_CONIC` A conic optimization.
3	`MSK_PROBTYPE_GECO` General convex optimization.
5	`MSK_PROBTYPE_MIXED` General nonlinear constraints and conic constraints. This combination can not be solved by MOSEK.
1	`MSK_PROBTYPE_QO` The problem is a quadratic optimization problem.

I.34 Problem status keys

Value	Name Description
6	`MSK_PRO_STA_PRIM_AND_DUAL_INFEAS` The problem is primal and dual infeasible.
4	`MSK_PRO_STA_PRIM_INFEAS` The problem is primal infeasible.
7	`MSK_PRO_STA_ILL_POSED` The problem is ill-posed. For example, it may be primal and dual feasible but have a positive duality gap.
0	`MSK_PRO_STA_UNKNOWN` Unknown problem status.
2	`MSK_PRO_STA_PRIM_FEAS` The problem is primal feasible.
8	`MSK_PRO_STA_NEAR_PRIM_AND_DUAL_FEAS` The problem is at least nearly primal and dual feasible.
10	`MSK_PRO_STA_NEAR_DUAL_FEAS` The problem is at least nearly dual feasible.
11	`MSK_PRO_STA_PRIM_INFEAS_OR_UNBOUNDED` The problem is either primal infeasible or unbounded. This may occur for mixed integer problems.
1	`MSK_PRO_STA_PRIM_AND_DUAL_FEAS` The problem is primal and dual feasible.
5	`MSK_PRO_STA_DUAL_INFEAS`

continued on next page

continued from previous page

	The problem is dual infeasible.
9	`MSK_PRO_STA_NEAR_PRIM_FEAS` The problem is at least nearly primal feasible.
3	`MSK_PRO_STA_DUAL_FEAS` The problem is dual feasible.

I.35 Interpretation of quadratic terms in MPS files

Value	Name Description
0	`MSK_Q_READ_ADD` All elements in a Q matrix are assumed to belong to the lower triangular part. Duplicate elements in a Q matrix are added together.
1	`MSK_Q_READ_DROP_LOWER` All elements in the strict lower triangular part of the Q matrices are dropped.
2	`MSK_Q_READ_DROP_UPPER` All elements in the strict upper triangular part of the Q matrices are dropped.

I.36 Response codes

Value	Name Description
1218	`MSK_RES_ERR_PARAM_TYPE` The parameter type is invalid.
1268	`MSK_RES_ERR_INV_SKX` Invalid value in `skx`.
1203	`MSK_RES_ERR_INDEX_IS_TOO_SMALL` An index in an argument is too small.
803	`MSK_RES_WRN_PRESOLVE_BAD_PRECISION` The presolve estimates that the model is specified with insufficient precision.
1500	`MSK_RES_ERR_INV_PROBLEM` Invalid problem type. Probably a nonconvex problem has been specified.
2505	`MSK_RES_ERR_CANNOT_CLONE_NL` A task with a nonlinear function call-back cannot be cloned.
1551	`MSK_RES_ERR_MIO_NO_OPTIMIZER` No optimizer is available for the current class of integer optimization problems.

continued on next page

continued from previous page

3003	`MSK_RES_ERR_API_NL_DATA` None
4004	`MSK_RES_TRM_MIO_NEAR_ABS_GAP` The mixed-integer optimizer terminated because the near optimal absolute gap tolerance was satisfied.
1254	`MSK_RES_ERR_MUL_A_ELEMENT` An element in A is defined multiple times.
1170	`MSK_RES_ERR_INVALID_NAME_IN_SOL_FILE` An invalid name occurred in a solution file.
1114	`MSK_RES_ERR_MPS_MUL_QOBJ` The Q term in the objective is specified multiple times in the MPS data file.
1063	`MSK_RES_ERR_NO_INIT_ENV` `env` is not initialized.
1265	`MSK_RES_ERR_UNDEF_SOLUTION` The required solution is not defined.
1288	`MSK_RES_ERR_LASTJ` Invalid `lastj`.
1001	`MSK_RES_ERR_LICENSE_EXPIRED` The license has expired.
3055	`MSK_RES_ERR_SEN_INDEX_INVALID` Invalid range given in the sensitivity file.
1274	`MSK_RES_ERR_INV_SKN` Invalid value in `skn`.
1295	`MSK_RES_ERR_OBJ_Q_NOT_PSD` The quadratic coefficient matrix in the objective is not positive semidefinite as expected for a minimization problem.
1267	`MSK_RES_ERR_INV_SKC` Invalid value in `skc`.
1008	`MSK_RES_ERR_MISSING_LICENSE_FILE` MOSEK cannot find the license file or license server. Usually this happens if the operating system variable `MOSEKLM_LICENSE_FILE` is not set up appropriately. Please see the MOSEK installation manual for details.
1235	`MSK_RES_ERR_INDEX` An index is out of range.
3058	`MSK_RES_ERR_SEN_NUMERICAL` Numerical difficulties encountered performing the sensitivity analysis.
2800	`MSK_RES_ERR_LU_MAX_NUM_TRIES` Could not compute the LU factors of the matrix within the maximum number of allowed tries.
201	`MSK_RES_WRN_DROPPED_NZ_QOBJ` One or more non-zero elements were dropped in the Q matrix in the objective.

continued on next page

continued from previous page

3000	`MSK_RES_ERR_INTERNAL` An internal error occurred. Please report this problem.
800	`MSK_RES_WRN_NONCOMPLETE_LINEAR_DEPENDENCY_CHECK` The linear dependency check(s) was not completed and therefore the A matrix may contain linear dependencies.
1204	`MSK_RES_ERR_INDEX_IS_TOO_LARGE` An index in an argument is too large.
1154	`MSK_RES_ERR_LP_INVALID_VAR_NAME` A variable name is invalid when used in an LP formatted file.
2950	`MSK_RES_ERR_NO_DUAL_FOR_ITG_SOL` No dual information is available for the integer solution.
1150	`MSK_RES_ERR_LP_INCOMPATIBLE` The problem cannot be written to an LP formatted file.
1501	`MSK_RES_ERR_MIXED_PROBLEM` The problem contains both conic and nonlinear constraints.
1700	`MSK_RES_ERR_FEASREPAIR_CANNOT_RELAX` An optimization problem cannot be relaxed. This is the case e.g. for general nonlinear optimization problems.
1207	`MSK_RES_ERR_PARAM_NAME_INT` The parameter name is not correct for an integer parameter.
3057	`MSK_RES_ERR_SEN_SOLUTION_STATUS` No optimal solution found to the original problem given for sensitivity analysis.
1432	`MSK_RES_ERR_USER_NLO_FUNC` The user-defined nonlinear function reported an error.
405	`MSK_RES_WRN_TOO_MANY_BASIS_VARS` A basis with too many variables has been specified.
1081	`MSK_RES_ERR_SPACE_NO_INFO` No available information about the space usage.
1205	`MSK_RES_ERR_PARAM_NAME` The parameter name is not correct.
3056	`MSK_RES_ERR_SEN_INVALID_REGEXP` Syntax error in regexp or regexp longer than 1024.
200	`MSK_RES_WRN_NZ_IN_UPR_TRI` Non-zero elements specified in the upper triangle of a matrix were ignored.
505	`MSK_RES_WRN_LICENSE_FEATURE_EXPIRE` The license expires.
1263	`MSK_RES_ERR_NEGATIVE_SURPLUS` Negative surplus.
1404	`MSK_RES_ERR_INV_QCON_SUBK` Invalid value in `qcsubk`.
1406	`MSK_RES_ERR_INV_QCON_SUBJ` Invalid value in `qcsubj`.

continued on next page

continued from previous page

1198	`MSK_RES_ERR_ARGUMENT_TYPE` Incorrect argument type.
1017	`MSK_RES_ERR_LICENSE_MOSEKLM_DAEMON` The MOSEKLM license manager daemon is not up and running.
2901	`MSK_RES_ERR_INVALID_WCHAR` An invalid `wchar` string is encountered.
1059	`MSK_RES_ERR_END_OF_FILE` End of file reached.
1462	`MSK_RES_ERR_NAN_IN_BUC` u^c contains an invalid floating point value, i.e. a `NaN`.
1290	`MSK_RES_ERR_NONLINEAR_EQUALITY` The model contains a nonlinear equality which defines a nonconvex set.
1055	`MSK_RES_ERR_DATA_FILE_EXT` The data file format cannot be determined from the file name.
1210	`MSK_RES_ERR_PARAM_INDEX` Parameter index is out of range.
1285	`MSK_RES_ERR_FIRSTI` Invalid `firsti`.
1000	`MSK_RES_ERR_LICENSE` Invalid license.
1299	`MSK_RES_ERR_ARGUMENT_PERM_ARRAY` An invalid permutation array is specified.
85	`MSK_RES_WRN_LP_DROP_VARIABLE` Ignored a variable because the variable was not previously defined. Usually this implies that a variable appears in the bound section but not in the objective or the constraints.
1287	`MSK_RES_ERR_FIRSTJ` Invalid `firstj`.
1090	`MSK_RES_ERR_READ_FORMAT` The specified format cannot be read.
1219	`MSK_RES_ERR_INF_DOU_INDEX` A double information index is out of range for the specified type.
1286	`MSK_RES_ERR_LASTI` Invalid `lasti`.
1199	`MSK_RES_ERR_NR_ARGUMENTS` Incorrect number of function arguments.
1293	`MSK_RES_ERR_CON_Q_NOT_PSD` The quadratic constraint matrix is not positive semi-definite as expected for a constraint with finite upper bound. This results in a nonconvex problem.
63	`MSK_RES_WRN_ZERO_AIJ` One or more zero elements are specified in A.
2504	`MSK_RES_ERR_INV_NUMJ`

continued on next page

continued from previous page

	Invalid numj.
1650	`MSK_RES_ERR_FACTOR` An error occurred while factorizing a matrix.
3201	`MSK_RES_ERR_INVALID_BRANCH_PRIORITY` An invalid branching priority is specified. It should be nonnegative.
1216	`MSK_RES_ERR_PARAM_IS_TOO_SMALL` The parameter value is too small.
1163	`MSK_RES_ERR_LP_WRITE_CONIC_PROBLEM` The problem contains cones that cannot be written to an LP formatted file.
4000	`MSK_RES_TRM_MAX_ITERATIONS` The optimizer terminated at the maximum number of iterations.
1240	`MSK_RES_ERR_MAXNUMCON` The maximum number of constraints specified is smaller than the number of constraints in the task.
1050	`MSK_RES_ERR_UNKNOWN` Unknown error.
1162	`MSK_RES_ERR_READ_LP_NONEXISTING_NAME` A variable never occurred in objective or constraints.
2503	`MSK_RES_ERR_INV_NUMI` Invalid numi.
1292	`MSK_RES_ERR_NONLINEAR_RANGED` The model contains a nonlinear ranged constraint which by definition defines a nonconvex set.
1047	`MSK_RES_ERR_THREAD_MUTEX_UNLOCK` Could not unlock a mutex.
1100	`MSK_RES_ERR_MPS_FILE` An error occurred while reading an MPS file.
1156	`MSK_RES_ERR_WRITE_OPF_INVALID_VAR_NAME` Empty variable names cannot be written to OPF files.
1152	`MSK_RES_ERR_LP_DUP_SLACK_NAME` The name of the slack variable added to a ranged constraint already exists.
2000	`MSK_RES_ERR_NO_PRIMAL_INFEAS_CER` A certificate of primal infeasibility is not available.
1158	`MSK_RES_ERR_WRITE_LP_FORMAT` Problem cannot be written as an LP file.
3052	`MSK_RES_ERR_SEN_INDEX_RANGE` Index out of range in the sensitivity analysis file.
66	`MSK_RES_WRN_SPAR_MAX_LEN` A value for a string parameter is longer than the buffer that is supposed to hold it.
3050	`MSK_RES_ERR_SEN_FORMAT` Syntax error in sensitivity analysis file.

continued on next page

continued from previous page

1407	`MSK_RES_ERR_INV_QCON_VAL` Invalid value in `qcval`.
1206	`MSK_RES_ERR_PARAM_NAME_DOU` The parameter name is not correct for a double parameter.
1291	`MSK_RES_ERR_NONCONVEX` The optimization problem is nonconvex.
1300	`MSK_RES_ERR_CONE_INDEX` An index of a non-existing cone has been specified.
1470	`MSK_RES_ERR_NAN_IN_C` c contains an invalid floating point value, i.e. a `NaN`.
1304	`MSK_RES_ERR_MAXNUMCONE` The value specified for `maxnumcone` is too small.
1103	`MSK_RES_ERR_MPS_NULL_CON_NAME` An empty constraint name is used in an MPS file.
1417	`MSK_RES_ERR_QCON_UPPER_TRIANGLE` An element in the upper triangle of a Q^k is specified. Only elements in the lower triangle should be specified.
4015	`MSK_RES_TRM_NUM_MAX_NUM_INT_SOLUTIONS` The mixed-integer optimizer terminated as the maximum number of feasible solutions was reached.
1125	`MSK_RES_ERR_MPS_TAB_IN_FIELD2` A tab char occurred in field 2.
270	`MSK_RES_WRN_MIO_INFEASIBLE_FINAL` The final mixed integer problem with all the integer variables fixed at their optimal values is infeasible.
1251	`MSK_RES_ERR_NUMVARLIM` Maximum number of variables limit is exceeded.
1433	`MSK_RES_ERR_USER_NLO_EVAL` The user-defined nonlinear function reported an error.
1232	`MSK_RES_ERR_INF_TYPE` The information type is invalid.
1106	`MSK_RES_ERR_MPS_UNDEF_VAR_NAME` An undefined variable name occurred in an MPS file.
503	`MSK_RES_WRN_USING_GENERIC_NAMES` The file writer reverts to generic names because a name is blank.
1127	`MSK_RES_ERR_MPS_TAB_IN_FIELD5` A tab char occurred in field 5.
1056	`MSK_RES_ERR_INVALID_FILE_NAME` An invalid file name has been specified.
804	`MSK_RES_WRN_WRITE_DISCARDED_CFIX` The fixed objective term could not be converted to a variable and was discarded in the output file.
1415	`MSK_RES_ERR_QOBJ_UPPER_TRIANGLE`

continued on next page

continued from previous page

	An element in the upper triangle of Q^o is specified. Only elements in the lower triangle should be specified.
1054	`MSK_RES_ERR_FILE_WRITE` File write error.
1164	`MSK_RES_ERR_LP_WRITE_GECO_PROBLEM` The problem contains general convex terms that cannot be written to an LP formatted file.
1243	`MSK_RES_ERR_MAXNUMQNZ` The maximum number of non-zeros specified for the Q matrices is smaller than the number of non-zeros in the current Q matrices.
2506	`MSK_RES_ERR_CANNOT_HANDLE_NL` A function cannot handle a task with nonlinear function call-backs.
1600	`MSK_RES_ERR_NO_BASIS_SOL` No basic solution is defined.
1131	`MSK_RES_ERR_ORD_INVALID` Invalid content in branch ordering file.
1303	`MSK_RES_ERR_CONE_REP_VAR` A variable is included multiple times in the cone.
1075	`MSK_RES_ERR_INVALID_OBJ_NAME` An invalid objective name is specified.
1052	`MSK_RES_ERR_FILE_OPEN` Error while opening a file.
250	`MSK_RES_WRN_IGNORE_INTEGER` Ignored integer constraints.
1296	`MSK_RES_ERR_OBJ_Q_NOT_NSD` The quadratic coefficient matrix in the objective is not negative semidefinite as expected for a maximization problem.
1064	`MSK_RES_ERR_INVALID_TASK` The `task` is invalid.
1065	`MSK_RES_ERR_NULL_POINTER` An argument to a function is unexpectedly a NULL pointer.
3005	`MSK_RES_ERR_API_FATAL_ERROR` An internal error occurred in the API. Please report this problem.
1550	`MSK_RES_ERR_INV_OPTIMIZER` An invalid optimizer has been chosen for the problem. This means that the simplex or the conic optimizer is chosen to optimize a nonlinear problem.
1310	`MSK_RES_ERR_REMOVE_CONE_VARIABLE` A variable cannot be removed because it will make a cone invalid.
62	`MSK_RES_WRN_LARGE_AIJ` A numerically large value is specified for one $a_{i,j}$.
1208	`MSK_RES_ERR_PARAM_NAME_STR` The parameter name is not correct for a string parameter.
1018	`MSK_RES_ERR_LICENSE_FEATURE`

continued on next page

continued from previous page

	A requested feature is not available in the license file(s). Most likely due to an incorrect license system setup.
251	`MSK_RES_WRN_NO_GLOBAL_OPTIMIZER` No global optimizer is available.
1040	`MSK_RES_ERR_LINK_FILE_DLL` A file cannot be linked to a stream in the DLL version.
1701	`MSK_RES_ERR_FEASREPAIR_SOLVING_RELAXED` The relaxed problem could not be solved to optimality. Please consult the log file for further details.
1221	`MSK_RES_ERR_INDEX_ARR_IS_TOO_SMALL` An index in an array argument is too small.
1259	`MSK_RES_ERR_SOLVER_PROBTYPE` Problem type does not match the chosen optimizer.
1220	`MSK_RES_ERR_INF_INT_INDEX` An integer information index is out of range for the specified type.
1053	`MSK_RES_ERR_FILE_READ` File read error.
1440	`MSK_RES_ERR_USER_NLO_EVAL_HESSUBI` The user-defined nonlinear function reported an invalid subscript in the Hessian.
1441	`MSK_RES_ERR_USER_NLO_EVAL_HESSUBJ` The user-defined nonlinear function reported an invalid subscript in the Hessian.
300	`MSK_RES_WRN_SOL_FILTER` Invalid solution filter is specified.
3100	`MSK_RES_ERR_UNB_STEP_SIZE` A step size in an optimizer was unexpectedly unbounded. For instance, if the step-size becomes unbounded in phase 1 of the simplex algorithm then an error occurs. Normally this will happen only if the problem is badly formulated. Please contact MOSEK support if this error occurs.
4030	`MSK_RES_TRM_INTERNAL` The optimizer terminated due to some internal reason. Please contact MOSEK support.
1110	`MSK_RES_ERR_MPS_NO_OBJECTIVE` No objective is defined in an MPS file.
1400	`MSK_RES_ERR_INFINITE_BOUND` A finite bound value is too large in absolute value.
1030	`MSK_RES_ERR_OPEN_DL` A dynamic link library could not be opened.
3001	`MSK_RES_ERR_API_ARRAY_TOO_SMALL` An input array was too short.
1046	`MSK_RES_ERR_THREAD_MUTEX_LOCK` Could not lock a mutex.

continued on next page

continued from previous page

1262	`MSK_RES_ERR_LAST` Invalid `last`.
1151	`MSK_RES_ERR_LP_EMPTY` The problem cannot be written to an LP formatted file.
1011	`MSK_RES_ERR_SIZE_LICENSE_VAR` The problem has too many variables to be solved with the available license.
1062	`MSK_RES_ERR_INVALID_STREAM` An invalid stream is referenced.
2520	`MSK_RES_ERR_INVALID_ACCMODE` An invalid access mode is specified.
1250	`MSK_RES_ERR_NUMCONLIM` Maximum number of constraints limit is exceeded.
1104	`MSK_RES_ERR_MPS_NULL_VAR_NAME` An empty variable name is used in an MPS file.
72	`MSK_RES_WRN_MPS_SPLIT_BOU_VECTOR` A BOUNDS vector is split into several nonadjacent parts in an MPS file.
1026	`MSK_RES_ERR_LICENSE_SERVER_VERSION` The version specified in the checkout request is greater than the highest version number the daemon supports.
1025	`MSK_RES_ERR_LICENSE_INVALID_HOSTID` The host ID specified in the license file does not match the host ID of the computer.
1045	`MSK_RES_ERR_THREAD_MUTEX_INIT` Could not initialize a mutex.
1280	`MSK_RES_ERR_INV_NAME_ITEM` An invalid name item code is used.
53	`MSK_RES_WRN_LARGE_UP_BOUND` A large but finite upper bound in absolute value has been specified.
4009	`MSK_RES_TRM_MIO_NUM_BRANCHES` The mixed-integer optimizer terminated as to the maximum number of branches was reached.
1272	`MSK_RES_ERR_INV_CONE_TYPE` Invalid cone type code is encountered.
1112	`MSK_RES_ERR_MPS_MUL_CON_NAME` A constraint name was specified multiple times in the `ROWS` section.
1801	`MSK_RES_ERR_INVALID_IOMODE` Invalid io mode.
1115	`MSK_RES_ERR_MPS_INV_SEC_ORDER` The sections in the MPS data file are not in the correct order.
1016	`MSK_RES_ERR_LICENSE_MAX` Maximum number of licenses is reached.
4007	`MSK_RES_TRM_USER_CALLBACK`

continued on next page

continued from previous page

	The optimizer terminated due to the return of the user-defined call-back function.
1430	`MSK_RES_ERR_USER_FUNC_RET` An user function reported an error.
1058	`MSK_RES_ERR_INVALID_MBT_FILE` A MOSEK binary task file is invalid.
1294	`MSK_RES_ERR_CON_Q_NOT_NSD` The quadratic constraint matrix is not negative semi-definite as expected for a constraint with finite lower bound. This results in a nonconvex problem.
3600	`MSK_RES_ERR_XML_INVALID_PROBLEM_TYPE` The problem type is not supported by the XML format.
1231	`MSK_RES_ERR_INF_INT_NAME` A integer information name is invalid.
1107	`MSK_RES_ERR_MPS_INV_CON_KEY` An invalid constraint key occurred in an MPS file.
2001	`MSK_RES_ERR_NO_DUAL_INFEAS_CER` A certificate of infeasibility is not available.
4025	`MSK_RES_TRM_NUMERICAL_PROBLEM` The optimizer terminated due to numerical problems.
52	`MSK_RES_WRN_LARGE_LO_BOUND` A large but finite lower bound in absolute value has been specified.
3999	`MSK_RES_ERR_API_INTERNAL` None
70	`MSK_RES_WRN_MPS_SPLIT_RHS_VECTOR` An RHS vector is split into several nonadjacent parts in an MPS file.
3053	`MSK_RES_ERR_SEN_BOUND_INVALID_UP` Analysis of upper bound requested for an index, where no upper bound exists.
1702	`MSK_RES_ERR_FEASREPAIR_INCONSISTENT_BOUND` The upper bound is less than the lower bound for a variable or a constraint. Please correct this before running the feasibility repair.
1449	`MSK_RES_ERR_Y_IS_UNDEFINED` The solution item y is undefined.
3200	`MSK_RES_ERR_INVALID_BRANCH_DIRECTION` An invalid branching direction is specified.
3004	`MSK_RES_ERR_API_CALLBACK` None
1305	`MSK_RES_ERR_CONE_TYPE` Invalid cone type specified.
4008	`MSK_RES_TRM_MIO_NUM_RELAXS` The mixed-integer optimizer terminated as the maximum number of relaxations was reached.
1256	`MSK_RES_ERR_INV_BKC`

continued on next page

continued from previous page

	Invalid bound key is specified for a constraint.
4020	`MSK_RES_TRM_MAX_NUM_SETBACKS` The optimizer terminated as the maximum number of set-backs was reached. This indicates numerical problems and a possibly badly formulated problem.
3101	`MSK_RES_ERR_IDENTICAL_TASKS` Some tasks related to this function call were identical. Unique tasks were expected.
1020	`MSK_RES_ERR_LICENSE_CANNOT_ALLOCATE` The license system cannot allocate the memory required.
1402	`MSK_RES_ERR_INV_QOBJ_SUBJ` Invalid value in `qosubj`.
1302	`MSK_RES_ERR_CONE_OVERLAP` A new cone which variables overlap with an existing cone has been specified.
1401	`MSK_RES_ERR_INV_QOBJ_SUBI` Invalid value in `qosubi`.
1153	`MSK_RES_ERR_WRITE_MPS_INVALID_NAME` An invalid name is created while writing an MPS file. Usually this will make the MPS file unreadable.
1553	`MSK_RES_ERR_MIO_NOT_LOADED` The mixed-integer optimizer is not loaded.
1061	`MSK_RES_ERR_NULL_TASK` `task` is a NULL pointer.
1450	`MSK_RES_ERR_NAN_IN_DOUBLE_DATA` An invalid floating point value was used in some double data.
3059	`MSK_RES_ERR_CONCURRENT_OPTIMIZER` An unsupported optimizer was chosen for use with the concurrent optimizer.
1252	`MSK_RES_ERR_TOO_SMALL_MAXNUMANZ` Maximum number of non-zeros allowed in A is too small.
1197	`MSK_RES_ERR_ARGUMENT_LENNEQ` Incorrect length of arguments.
500	`MSK_RES_WRN_LICENSE_EXPIRE` The license expires.
1200	`MSK_RES_ERR_IN_ARGUMENT` A function argument is incorrect.
1051	`MSK_RES_ERR_SPACE` Out of space.
1241	`MSK_RES_ERR_MAXNUMVAR` The maximum number of variables specified is smaller than the number of variables in the task.
1800	`MSK_RES_ERR_INVALID_COMPRESSION` Invalid compression type.

continued on next page

continued from previous page

1101	`MSK_RES_ERR_MPS_INV_FIELD` A field in the MPS file is invalid. Probably it is too wide.
1060	`MSK_RES_ERR_NULL_ENV` `env` is a NULL pointer.
3500	`MSK_RES_ERR_INTERNAL_TEST_FAILED` An internal unit test function failed.
501	`MSK_RES_WRN_LICENSE_SERVER` The license server is not responding.
1122	`MSK_RES_ERR_MPS_INVALID_OBJSENSE` An invalid objective sense is specified.
1168	`MSK_RES_ERR_OPF_FORMAT` Syntax error in an OPF file
1269	`MSK_RES_ERR_INV_SK_STR` Invalid status key string encountered.
1071	`MSK_RES_ERR_DUP_NAME` An error occurred while reading an MPS file..
1116	`MSK_RES_ERR_MPS_MUL_CSEC` Multiple `CSECTION`s are given the same name.
51	`MSK_RES_WRN_LARGE_BOUND` A very large bound in absolute value has been specified.
50	`MSK_RES_WRN_OPEN_PARAM_FILE` The parameter file could not be opened.
1431	`MSK_RES_ERR_USER_FUNC_RET_DATA` An user function returned invalid data.
1615	`MSK_RES_ERR_BASIS_SINGULAR` The basis is singular and hence cannot be factored.
1155	`MSK_RES_ERR_LP_FREE_CONSTRAINT` Free constraints cannot be written in LP file format.
1445	`MSK_RES_ERR_INVALID_OBJECTIVE_SENSE` An invalid objective sense is specified.
0	`MSK_RES_OK` No error occurred.
3002	`MSK_RES_ERR_API_CB_CONNECT` None
1253	`MSK_RES_ERR_INV_APTRE` `aptre[j]` is strictly smaller than `aptrb[j]` for some `j`.
1013	`MSK_RES_ERR_OPTIMIZER_LICENSE` The optimizer required is not licensed.
1007	`MSK_RES_ERR_FILE_LICENSE` Invalid license file.
1160	`MSK_RES_ERR_LP_FORMAT` Syntax error in an LP file.
1237	`MSK_RES_ERR_SOLITEM`

continued on next page

continued from previous page

	The solution item number `solitem` is invalid. Please note `MSK_SOL_ITEM_SNX` is invalid for the basic solution.
1010	`MSK_RES_ERR_SIZE_LICENSE_CON` The problem has too many constraints to be solved with the available license.
1118	`MSK_RES_ERR_MPS_CONE_OVERLAP` A variable is specified to be a member of several cones.
700	`MSK_RES_WRN_ZEROS_IN_SPARSE_DATA` One or more almost zero elements are specified in sparse input data.
1408	`MSK_RES_ERR_QCON_SUBI_TOO_SMALL` Invalid value in `qcsubi`.
1610	`MSK_RES_ERR_BASIS_FACTOR` The factorization of the basis is invalid.
1580	`MSK_RES_ERR_POSTSOLVE` An error occurred during the postsolve. Please contact MOSEK support.
1215	`MSK_RES_ERR_PARAM_IS_TOO_LARGE` The parameter value is too large.
1281	`MSK_RES_ERR_PRO_ITEM` An invalid problem is used.
1057	`MSK_RES_ERR_INVALID_SOL_FILE_NAME` An invalid file name has been specified.
1271	`MSK_RES_ERR_INV_CONE_TYPE_STR` Invalid cone type string encountered.
1283	`MSK_RES_ERR_INVALID_FORMAT_TYPE` Invalid format type.
57	`MSK_RES_WRN_LARGE_CJ` A numerically large value is specified for one c_j.
1035	`MSK_RES_ERR_OLDER_DLL` The dynamic link library is older than the specified version.
1019	`MSK_RES_ERR_PLATFORM_NOT_LICENSED` A requested license feature is not available for the required platform.
1119	`MSK_RES_ERR_MPS_CONE_REPEAT` A variable is repeated within the `CSECTION`.
3051	`MSK_RES_ERR_SEN_UNDEF_NAME` An undefined name was encountered in the sensitivity analysis file.
1403	`MSK_RES_ERR_INV_QOBJ_VAL` Invalid value in `qoval`.
71	`MSK_RES_WRN_MPS_SPLIT_RAN_VECTOR` A RANGE vector is split into several nonadjacent parts in an MPS file.
3054	`MSK_RES_ERR_SEN_BOUND_INVALID_LO` Analysis of lower bound requested for an index, where no lower bound exists.

continued on next page

continued from previous page

1111	`MSK_RES_ERR_MPS_SPLITTED_VAR` A variable is split in an MPS data file.
1080	`MSK_RES_ERR_SPACE_LEAKING` MOSEK is leaking memory. This can be due to either an incorrect use of MOSEK or a bug.
1201	`MSK_RES_ERR_ARGUMENT_DIMENSION` A function argument is of incorrect dimension.
280	`MSK_RES_WRN_FIXED_BOUND_VALUES` A fixed constraint/variable has been specified using the bound keys but the numerical bounds are different. The variable is fixed at the lower bound.
4001	`MSK_RES_TRM_MAX_TIME` The optimizer terminated at the maximum amount of time.
1461	`MSK_RES_ERR_NAN_IN_BLC` l^c contains an invalid floating point value, i.e. a `NaN`.
1350	`MSK_RES_ERR_SOL_FILE_NUMBER` An invalid number is specified in a solution file.
3700	`MSK_RES_ERR_INVALID_AMPL_STUB` Invalid AMPL stub.
1260	`MSK_RES_ERR_OBJECTIVE_RANGE` Empty objective range.
1238	`MSK_RES_ERR_WHICHITEM_NOT_ALLOWED` `whichitem` is unacceptable.
1471	`MSK_RES_ERR_NAN_IN_BLX` l^x contains an invalid floating point value, i.e. a `NaN`.
1236	`MSK_RES_ERR_WHICHSOL` The solution defined by compwhichsol does not exists.
1242	`MSK_RES_ERR_MAXNUMANZ` The maximum number of non-zeros specified for A is smaller than the number of non-zeros in the current A.
801	`MSK_RES_WRN_ELIMINATOR_SPACE` The eliminator is skipped at least once due to lack of space.
1049	`MSK_RES_ERR_THREAD_COND_INIT` Could not initialize a condition.
1405	`MSK_RES_ERR_INV_QCON_SUBI` Invalid value in `qcsubi`.
1036	`MSK_RES_ERR_NEWER_DLL` The dynamic link library is newer than the specified version.
1409	`MSK_RES_ERR_QCON_SUBI_TOO_LARGE` Invalid value in `qcsubi`.
1113	`MSK_RES_ERR_MPS_MUL_QSEC` Multiple `QSECTION`s are specified for a constraint in the MPS data file.
502	`MSK_RES_WRN_EMPTY_NAME`

continued on next page

continued from previous page

	A variable or constraint name is empty. The output file may be invalid.
4003	`MSK_RES_TRM_MIO_NEAR_REL_GAP` The mixed-integer optimizer terminated because the near optimal relative gap tolerance was satisfied.
80	`MSK_RES_WRN_LP_OLD_QUAD_FORMAT` Missing '/2' after quadratic expressions in bound or objective.
1102	`MSK_RES_ERR_MPS_INV_MARKER` An invalid marker has been specified in the MPS file.
1070	`MSK_RES_ERR_NULL_NAME` An all blank name has been specified.
1264	`MSK_RES_ERR_NEGATIVE_APPEND` Cannot append a negative number.
1270	`MSK_RES_ERR_INV_SK` Invalid status key code.
1006	`MSK_RES_ERR_PROB_LICENSE` The software is not licensed to solve the problem.
1015	`MSK_RES_ERR_LICENSE_SERVER` The license server is not responding.
4005	`MSK_RES_TRM_USER_BREAK` The optimizer terminated due to a user break.
400	`MSK_RES_WRN_TOO_FEW_BASIS_VARS` An incomplete basis has been specified. Too few basis variables are specified.
1161	`MSK_RES_ERR_WRITE_LP_NON_UNIQUE_NAME` An auto-generated name is not unique.
1108	`MSK_RES_ERR_MPS_INV_BOUND_KEY` An invalid bound key occurred in an MPS file.
1472	`MSK_RES_ERR_NAN_IN_BUX` u^x contains an invalid floating point value, i.e. a `NaN`.
1109	`MSK_RES_ERR_MPS_INV_SEC_NAME` An invalid section name occurred in an MPS file.
1266	`MSK_RES_ERR_BASIS` An invalid basis is specified. Either too many or too few basis variables are specified.
1257	`MSK_RES_ERR_INV_BKX` An invalid bound key is specified for a variable.
1002	`MSK_RES_ERR_LICENSE_VERSION` The license is valid for another version of MOSEK.
4006	`MSK_RES_TRM_STALL`

continued on next page

continued from previous page

	The optimizer terminated due to slow progress. Normally there are three possible reasons for this: Either a bug in MOSEK, the problem is badly formulated, or, in case of nonlinear problems, the nonlinear call-back functions are incorrect. Please contact MOSEK support if this happens.
1048	`MSK_RES_ERR_THREAD_CREATE` Could not create a thread. This error may occur if a large number of environments are created and not deleted again. In any case it is a good practice to minimize the number of environments created.
1128	`MSK_RES_ERR_MPS_INVALID_OBJ_NAME` An invalid objective name is specified.
1217	`MSK_RES_ERR_PARAM_VALUE_STR` The parameter value string is incorrect.
1222	`MSK_RES_ERR_INDEX_ARR_IS_TOO_LARGE` An index in an array argument is too large.
1306	`MSK_RES_ERR_CONE_TYPE_STR` Invalid cone type specified.
1301	`MSK_RES_ERR_CONE_SIZE` A cone with too few members is specified.
1258	`MSK_RES_ERR_INV_VAR_TYPE` An invalid variable type is specified for a variable.
1157	`MSK_RES_ERR_LP_FILE_FORMAT` Syntax error in an LP file.
1021	`MSK_RES_ERR_LICENSE_CANNOT_CONNECT` MOSEK cannot connect to the license server. Most likely the license server is not up and running.
4002	`MSK_RES_TRM_OBJECTIVE_RANGE` The optimizer terminated on the bound of the objective range.
1126	`MSK_RES_ERR_MPS_TAB_IN_FIELD3` A tab char occurred in field 3.
350	`MSK_RES_WRN_UNDEF_SOL_FILE_NAME` Undefined name occurred in a solution.
1255	`MSK_RES_ERR_INV_BK` Invalid bound key.
1014	`MSK_RES_ERR_FLEXLM` The FLEXlm license manager reported an error.
2550	`MSK_RES_ERR_MBT_INCOMPATIBLE` The MBT file is incompatible with this platform. This results from reading a file on a 32 bit platform generated on a 64 bit platform.
65	`MSK_RES_WRN_NAME_MAX_LEN` A name is longer than the buffer that is supposed to hold it.
1165	`MSK_RES_ERR_NAME_MAX_LEN` A name is longer than the buffer that is supposed to hold it.
1261	`MSK_RES_ERR_FIRST`

continued on next page

continued from previous page

	Invalid `first`.
4031	`MSK_RES_TRM_INTERNAL_STOP` The optimizer terminated for internal reasons. Please contact MOSEK support.
1117	`MSK_RES_ERR_MPS_CONE_TYPE` Invalid cone type specified in a `CSECTION`.
1005	`MSK_RES_ERR_SIZE_LICENSE` The problem is bigger than the license.
1375	`MSK_RES_ERR_HUGE_C` A huge value in absolute size is specified for one c_j.
1446	`MSK_RES_ERR_UNDEFINED_OBJECTIVE_SENSE` The objective sense has not been specified before the optimization.
802	`MSK_RES_WRN_PRESOLVE_OUTOFSPACE` The presolve is incomplete due to lack of space.
1130	`MSK_RES_ERR_ORD_INVALID_BRANCH_DIR` An invalid branch direction key is specified.
2501	`MSK_RES_ERR_INV_MARKI` Invalid value in marki.
2502	`MSK_RES_ERR_INV_MARKJ` Invalid value in markj.
2500	`MSK_RES_ERR_NO_SOLUTION_IN_CALLBACK` The required solution is not available.
2900	`MSK_RES_ERR_INVALID_UTF8` An invalid UTF8 string is encountered.
1105	`MSK_RES_ERR_MPS_UNDEF_CON_NAME` An undefined constraint name occurred in an MPS file.
1012	`MSK_RES_ERR_SIZE_LICENSE_INTVAR` The problem contains too many integer variables to be solved with the available license.
1230	`MSK_RES_ERR_INF_DOU_NAME` A double information name is invalid.
1552	`MSK_RES_ERR_NO_OPTIMIZER_VAR_TYPE` No optimizer is available for this class of optimization problems.

I.37 Response code type

Value	Name Description
1	`MSK_RESPONSE_WRN` The response code is a warning.
2	`MSK_RESPONSE_TRM` The response code is an optimizer termination status.
4	`MSK_RESPONSE_UNK`

continued on next page

continued from previous page

	The response code does not belong to any class.
0	`MSK_RESPONSE_OK`
	The response code is OK.
3	`MSK_RESPONSE_ERR`
	The response code is an error.

I.38 Scaling type

Value	Name Description
1	`MSK_SCALING_NONE`
	No scaling is performed.
2	`MSK_SCALING_MODERATE`
	A conservative scaling is performed.
3	`MSK_SCALING_AGGRESSIVE`
	A very aggressive scaling is performed.
0	`MSK_SCALING_FREE`
	The optimizer chooses the scaling heuristic.

I.39 Sensitivity types

Value	Name Description
1	`MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION`
	Optimal partition sensitivity analysis is performed.
0	`MSK_SENSITIVITY_TYPE_BASIS`
	Basis sensitivity analysis is performed.

I.40 Degeneracy strategies

Value	Name Description
0	`MSK_SIM_DEGEN_NONE`
	The simplex optimizer should use no degeneration strategy.
3	`MSK_SIM_DEGEN_MODERATE`
	The simplex optimizer should use a moderate degeneration strategy.
4	`MSK_SIM_DEGEN_MINIMUM`
	The simplex optimizer should use a minimum degeneration strategy.
2	`MSK_SIM_DEGEN_AGGRESSIVE`

continued on next page

continued from previous page

Value	Name Description
	The simplex optimizer should use an aggressive degeneration strategy.
1	`MSK_SIM_DEGEN_FREE` The simplex optimizer chooses the degeneration strategy.

I.41 Hot-start type employed by the simplex optimizer

Value	Name Description
0	`MSK_SIM_HOTSTART_NONE` The simplex optimizer performs a coldstart.
2	`MSK_SIM_HOTSTART_STATUS_KEYS` Only the status keys of the constraints and variables are used to choose the type of hot-start.
1	`MSK_SIM_HOTSTART_FREE` The simplex optimize chooses the hot-start type.

I.42 Simplex selection strategy

Value	Name Description
1	`MSK_SIM_SELECTION_FULL` The optimizer uses full pricing.
5	`MSK_SIM_SELECTION_PARTIAL` The optimizer uses a partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.
0	`MSK_SIM_SELECTION_FREE` The optimizer chooses the pricing strategy.
2	`MSK_SIM_SELECTION_ASE` The optimizer uses approximate steepest-edge pricing.
3	`MSK_SIM_SELECTION_DEVEX` The optimizer uses devex steepest-edge pricing (or if it is not available an approximate steep-edge selection).
4	`MSK_SIM_SELECTION_SE` The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

I.43 Solution items

Value	Name Description
4	`MSK_SOL_ITEM_SUC` Lagrange multipliers for upper bounds on the constraints.
0	`MSK_SOL_ITEM_XC` Solution for the constraints.
1	`MSK_SOL_ITEM_XX` Variable solution.
2	`MSK_SOL_ITEM_Y` Lagrange multipliers for equations.
5	`MSK_SOL_ITEM_SLX` Lagrange multipliers for lower bounds on the variables.
6	`MSK_SOL_ITEM_SUX` Lagrange multipliers for upper bounds on the variables.
7	`MSK_SOL_ITEM_SNX` Lagrange multipliers corresponding to the conic constraints on the variables.
3	`MSK_SOL_ITEM_SLC` Lagrange multipliers for lower bounds on the constraints.

I.44 Solution status keys

Value	Name Description
6	`MSK_SOL_STA_DUAL_INFEAS_CER` The solution is a certificate of dual infeasibility.
5	`MSK_SOL_STA_PRIM_INFEAS_CER` The solution is a certificate of primal infeasibility.
0	`MSK_SOL_STA_UNKNOWN` Status of the solution is unknown.
8	`MSK_SOL_STA_NEAR_OPTIMAL` The solution is nearly optimal.
12	`MSK_SOL_STA_NEAR_PRIM_INFEAS_CER` The solution is almost a certificate of primal infeasibility.
2	`MSK_SOL_STA_PRIM_FEAS` The solution is primal feasible.
15	`MSK_SOL_STA_NEAR_INTEGER_OPTIMAL` The primal solution is near integer optimal.
10	`MSK_SOL_STA_NEAR_DUAL_FEAS` The solution is nearly dual feasible.
14	`MSK_SOL_STA_INTEGER_OPTIMAL` The primal solution is integer optimal.
13	`MSK_SOL_STA_NEAR_DUAL_INFEAS_CER` The solution is almost a certificate of dual infeasibility.

continued on next page

continued from previous page

11	`MSK_SOL_STA_NEAR_PRIM_AND_DUAL_FEAS` The solution is nearly both primal and dual feasible.
1	`MSK_SOL_STA_OPTIMAL` The solution is optimal.
4	`MSK_SOL_STA_PRIM_AND_DUAL_FEAS` The solution is both primal and dual feasible.
9	`MSK_SOL_STA_NEAR_PRIM_FEAS` The solution is nearly primal feasible.
3	`MSK_SOL_STA_DUAL_FEAS` The solution is dual feasible.

I.45 Solution types

Value	Name Description
2	`MSK_SOL_ITG` The integer solution.
0	`MSK_SOL_ITR` The interior solution.
1	`MSK_SOL_BAS` The basic solution.

I.46 Solve primal or dual form

Value	Name Description
1	`MSK_SOLVE_PRIMAL` The optimizer should solve the primal problem.
2	`MSK_SOLVE_DUAL` The optimizer should solve the dual problem.
0	`MSK_SOLVE_FREE` The optimizer is free to solve either the primal or the dual problem.

I.47 String parameter types

Value	Name Description
8	`MSK_SPAR_PARAM_COMMENT_SIGN`

continued on next page

continued from previous page

	Only the first character in this string is used. It is considered as a start of comment sign in the MOSEK parameter file. Spaces are ignored in the string.
3	`MSK_SPAR_FEASREPAIR_NAME_PREFIX` Not applicable.
0	`MSK_SPAR_BAS_SOL_FILE_NAME` Name of the `bas` solution file.
12	`MSK_SPAR_READ_MPS_OBJ_NAME` Name of the free constraint used as objective function. An empty name means that the first constraint is used as objective function.
5	`MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL` The constraint and variable associated with the total weighted sum of violations are each given the name of this parameter postfixed with `CON` and `VAR` respectively.
4	`MSK_SPAR_FEASREPAIR_NAME_SEPARATOR` Not applicable.
10	`MSK_SPAR_PARAM_WRITE_FILE_NAME` The parameter database is written to this file.
6	`MSK_SPAR_INT_SOL_FILE_NAME` Name of the `int` solution file.
14	`MSK_SPAR_READ_MPS_RHS_NAME` Name of the RHS used. An empty name means that the first RHS vector is used.
21	`MSK_SPAR_STAT_FILE_NAME` Statistics file name.
24	`MSK_SPAR_WRITE_LP_GEN_VAR_NAME` Sometimes when an LP file is written additional variables must be inserted. They will have the prefix denoted by this parameter.
1	`MSK_SPAR_DATA_FILE_NAME` Data are read and written to this file.
13	`MSK_SPAR_READ_MPS_RAN_NAME` Name of the RANGE vector used. An empty name means that the first RANGE vector is used.
17	`MSK_SPAR_SOL_FILTER_XC_LOW` A filter used to determine which constraints should be listed in the solution file. A value of "0.5" means that all constraints having `xc[i]>0.5` should be listed, whereas "+0.5" means that all constraints having `xc[i]>=blc[i]+0.5` should be listed. An empty filter means that no filter is applied.
18	`MSK_SPAR_SOL_FILTER_XC_UPR`

continued on next page

continued from previous page

	A filter used to determine which constraints should be listed in the solution file. A value of “0.5” means that all constraints having `xc[i]<0.5` should be listed, whereas “-0.5” means all constraints having `xc[i]<=buc[i]-0.5` should be listed. An empty filter means that no filter is applied.
11	`MSK_SPAR_READ_MPS_BOU_NAME` Name of the BOUNDS vector used. An empty name means that the first BOUNDS vector is used.
20	`MSK_SPAR_SOL_FILTER_XX_UPR` A filter used to determine which variables should be listed in the solution file. A value of “0.5” means that all constraints having `xx[j]<0.5` should be printed, whereas “-0.5” means all constraints having `xx[j]<=bux[j]-0.5` should be listed. An empty filter means no filter is applied.
23	`MSK_SPAR_STAT_NAME` Name used when writing the statistics file.
9	`MSK_SPAR_PARAM_READ_FILE_NAME` Modifications to the parameter database is read from this file.
7	`MSK_SPAR_ITR_SOL_FILE_NAME` Name of the `itr` solution file.
15	`MSK_SPAR_SENSITIVITY_FILE_NAME` Not applicable.
2	`MSK_SPAR_DEBUG_FILE_NAME` MOSEK debug file.
22	`MSK_SPAR_STAT_KEY` Key used when writing the summary file.
16	`MSK_SPAR_SENSITIVITY_RES_FILE_NAME` Not applicable.
19	`MSK_SPAR_SOL_FILTER_XX_LOW` A filter used to determine which variables should be listed in the solution file. A value of “0.5” means that all constraints having `xx[j]>=0.5` should be listed, whereas “+0.5” means that all constraints having `xx[j]>=blx[j]+0.5` should be listed. An empty filter means no filter is applied.

I.48 Status keys

Value	Name Description
2	`MSK_SK_SUPBAS` The constraint or variable is super basic.

continued on next page

continued from previous page	
1	`MSK_SK_BAS` The constraint or variable is in the basis.
5	`MSK_SK_FIX` The constraint or variable is fixed.
3	`MSK_SK_LOW` The constraint or variable is at its lower bound.
6	`MSK_SK_INF` The constraint or variable is infeasible in the bounds.
0	`MSK_SK_UNK` The status for the constraint or variable is unknown.
4	`MSK_SK_UPR` The constraint or variable is at its upper bound.

I.49 Starting point types

Value	Name Description
1	`MSK_STARTING_POINT_CONSTANT` The starting point is set to a constant. This is more reliable than a non-constant starting point.
0	`MSK_STARTING_POINT_FREE` The starting point is chosen automatically.

I.50 Stream types

Value	Name Description
1	`MSK_STREAM_MSG` Message stream.
3	`MSK_STREAM_WRN` Warning stream.
0	`MSK_STREAM_LOG` Log stream.
2	`MSK_STREAM_ERR` Error stream.

I.51 Integer values

Value	Name Description
1024	`MSK_MAX_STR_LEN` Maximum string length allowed in MOSEK.
20	`MSK_LICENSE_BUFFER_LENGTH` The length of a license key buffer.

I.52 Variable types

Value	Name Description
1	`MSK_VAR_TYPE_INT` Is an integer variable.
0	`MSK_VAR_TYPE_CONT` Is a continuous variable.

I.53 XML writer output mode

Value	Name Description
1	`MSK_WRITE_XML_MODE_COL` Write in column order.
0	`MSK_WRITE_XML_MODE_ROW` Write in row order.

Bibliography

[1] Richard C. Grinold abd Ronald N. Kahn. *Active portfolio management*. McGraw-Hill, New York, 2 edition, 2000.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. Network flows. In G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, editors, *Optimization*, volume 1, pages 211–369. North Holland, Amsterdam, 1989.

[3] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Math. Programming*, 95(1):3–51, 2003.

[4] E. D. Andersen and K. D. Andersen. Presolving in linear programming. *Math. Programming*, 71(2):221–245, 1995.

[5] 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.

[6] E. D. Andersen, C. Roos, and T. Terlaky. On implementing a primal-dual interior-point method for conic quadratic optimization. *Math. Programming*, 95(2), February 2003.

[7] E. D. Andersen and Y. Ye. Combining interior-point and pivoting algorithms. *Management Sci.*, 42(12):1719–1731, December 1996.

[8] E. D. Andersen and Y. Ye. A computational study of the homogeneous algorithm for large-scale convex optimization. *Computational Optimization and Applications*, 10:243–269, 1998.

[9] E. D. Andersen and Y. Ye. On a homogeneous algorithm for the monotone complementarity problem. *Math. Programming*, 84(2):375–399, February 1999.

[10] K. D. Andersen. A Modified Schur Complement Method for Handling Dense Columns in Interior-Point Methods for Linear Programming. *ACM Trans. Math. Software*, 22(3):348–356, 1996.

[11] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming: Theory and algorithms*. John Wiley and Sons, New York, 2 edition, 1993.

[12] A. Ben-Tal and A Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. MPS/SIAM Series on Optimization. SIAM, 2001.

[13] V. Chvátal. *Linear programming*. W.H. Freeman and Company, 1983.

[14] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL. A modeling language for mathematical programming.* Duxbury Press, Belmont, CA, 1997.

[15] N. Gould and P. L. Toint. Preprocessing for quadratic programming. *Math. Programming*, 100(1):95–132, 2004.

[16] J. L. Kennington and K. R. Lewis. Generalized networks: The theory of preprocessing and an emperical analysis. *INFORMS Journal on Computing*, 16(2):162–173, 2004.

[17] M. S. Lobo, L. Vanderberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra Appl.*, 284:193–228, November 1998.

[18] M. S. Lobo, M. Fazel, and S. Boyd. Portfolio optimization with linear and fixed transaction costs. Technical report, CDS, California Institute of Technology, 2005. To appear in Annals of Operations Research. `http://www.cds.caltech.edu/~maryam/portfolio.html`.

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

[20] 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.

[21] Bernd Scherer. *Portfolio construction and risk budgeting.* Risk Books, 2 edition, 2004.

[22] R. J. Vanderbei. *Linear Programming. Foundations and Extensions.* Kluwer Academic Publishers, Boston/London/Dordrect, 1997.

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

[24] H. P. Williams. *Model building in mathematical programming.* John Wiley and Sons, 3 edition, 1993.

[25] L. A. Wolsey. *Integer programming.* John Wiley and Sons, 1998.

Index