The MOSEK optimization tools manual.
Version 6.0 (Revision 137).

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 at

`mosek\6\license.pdf`

Chapter 1

Changes and new features in MOSEK

The section presents improvements and new features added to MOSEK in version 6.0.

1.1 Compilers used to build MOSEK

MOSEK has been build with the compiler shown in Table 1.1.

Platform	C compiler
linux32x86	Intel C 11.0 (gcc 4.3, glibc 2.3.4)
linux64x86	Intel C 11.0 (gcc 4.3, glibc 2.3.4)
osx32x86	Intel C 11.1 (gcc 4.0)
osx64x86	Intel C 11.1 (gcc 4.0)
solaris32x86	Sun Studio 12
solaris64x86	Sun Studio 12
win32x86	Intel C 11.0 (VS 2005)
win64x86	Intel C 11.0 (VS 2005)

Table 1.1: Compiler version used to build MOSEK
.

1.2 General changes

- A problem analyzer is now available. It generates an simple report with of statisics and information about the optimization problem and relevant warnings about the problem formulation are included.
- A solution analyzer is now available.

- All timing measures are now wall clock times
- MOSEK employs version 1.2.3 of the zlib library.
- MOSEK employs version 11.6.1 of the FLEXnet licensing tools.
- The convexity of quadratic and quadratic constrained optimization is checked explicitly.
- On Windows all DLLs and EXEs are now signed.
- On all platforms the Jar files are signed.
- MOSEK no longer deals with ctrl-c. The user is responsible for terminating MOSEK in the callback.

1.3 Optimizers

1.3.1 Interior point optimizer

- The speed and stability of interior-point optimizer for linear problems has been improved.
- The speed and stability of the interior-point optimizer for conic problems has been improved. In particular, it is much better at dealing with primal or dual infeasible problems.

1.3.2 The simplex optimizers

- Presolve is now much more effective for simplex optimizers hot-starts.

1.3.3 Mixed-integer optimizer

- The stopping criteria for the mixed-integer optimizer have been changed to conform better with industry standards.

1.4 License system

- The license conditions have been relaxed, so that a license is shared among all tasks using a single environment. This means that running several optimizations in parallel will only consume one license, as long as the associated tasks share a single MOSEK environment. Please note this is NOT useful when using the MATLAB parallel toolbox.
- By default a license remains checked out for the lifetime of the environment. This behavior can be changed using the parameter `MSK_IPAR_CACHE_LICENSE`.
- Flexlm has been upgraded to version 11.6 from version 11.4.

1.5 Other changes

- The documentation has been improved.

1.6 Interfaces

- The AMPL interface has been augmented so it is possible to pass an initial (feasible) integer solution to mixed-integer optimizer.
- The AMPL interface is now capable of reading the constraint and variable names if they are avialable.

1.7 Platform changes

- MAC OSX on the PowerPC platform is no longer supported.
- Solaris on the SPARC platform is no longer supported.
- MAC OSX is supported on Intel 64 bit X86 i.e. osx64x86.
- Add support for MATLAB R2009b.

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\6\help\index.html`

in the distribution.

3.2 Additional reading

In this manual it is assumed that the reader is familiar with mathematics and in particular mathematical optimization. Some introduction to linear programming is found in books such as "Linear programming" by Chvátal [12] or "Computer Solution of Linear Programs" by Nazareth [18]. For more theoretical aspects see e.g. "Nonlinear programming: Theory and algorithms" by Bazaraa, Shetty, and Sherali [10]. Finally, the book "Model building in mathematical programming" by Williams [22] provides an excellent introduction to modeling 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 that 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

$$
\begin{array}{lrcl}
\text{minimize} & & & x_1^2 + 0.1x_2^2 + x_3^2 - x_1x_3 - x_2 \\
\text{subject to} & 1 & \leq & x_1 + x_2 + x_3, \\
 & & & x \geq 0.
\end{array} \tag{4.2}
$$

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

$$0.5x^TQx + c^Tx$$

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 converted 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, only the two types

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

and

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

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

Consider the problem

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

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

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

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 converted 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

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

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 [13] 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 9.4.

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.40 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 [13] for details about this option.

5.6 Constraint and variable names

AMPL assigns meaningfull names to all the constraints and variables. Since MOSEK uses item names in error and log messages, it may be useful to pass the AMPL names to MOSEK. Using the command

```
ampl: option mosek_auxfiles rc;
```

before the

```
solve;
```

command makes MOSEK obtain the constraint and variable names automatically.

5.7 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.

5.8 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: data 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:

```
Accepted: msk_ipar_optimizer                 = MSK_OPTIMIZER_PRIMAL_SIMPLEX
Accepted: outlev                             = 2

Computer   - Platform              : Linux/64-X86
Computer   - CPU type              : Intel-P4
MOSEK      - task name             :
MOSEK      - objective sense       : min
MOSEK      - problem type          : LO (linear optimization problem)
MOSEK      - constraints           : 7                   variables              : 9
MOSEK      - integer variables     : 0
Optimizer started.
Simplex optimizer started.
Presolve started.
Linear dependency checker started.
Linear dependency checker terminated.
Presolve   - Stk. size (kb) : 0
Eliminator - tries                 : 0                   time                   : 0.00
Eliminator - elim's                : 0
Lin. dep.  - tries                 : 1                   time                   : 0.00
```

```
Lin. dep.  - number               : 0
Presolve terminated. Time: 0.00
Primal simplex optimizer started.
Primal simplex optimizer setup started.
Primal simplex optimizer setup terminated.
Optimizer  - solved problem        : the primal
Optimizer  - constraints           : 7                 variables              : 9
Optimizer  - hotstart              : no
ITER      DEGITER(%)  PFEAS       DFEAS       POBJ                  DOBJ                  TIME      TOTTIME
0         0.00        1.40e+03    NA          1.2586666667e+01      NA                    0.00      0.01
3         0.00        0.00e+00    NA          1.4855737705e+01      NA                    0.00      0.01
Primal simplex optimizer terminated.
Simplex optimizer terminated. Time: 0.00.
Optimizer terminated. Time: 0.01
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
;
Accepted: msk_ipar_optimizer                  = MSK_OPTIMIZER_PRIMAL_SIMPLEX
Accepted: outlev                              = 2
Basic solution
Problem status  : UNKNOWN
Solution status : UNKNOWN
Primal - objective: 1.4855737705e+01   eq. infeas.: 3.97e+03 max bound infeas.: 2.00e+03
Dual   - objective: 0.0000000000e+00   eq. infeas.: 7.14e-01 max bound infeas.: 0.00e+00

Computer   - Platform              : Linux/64-X86
Computer   - CPU type              : Intel-P4
MOSEK      - task name             :
MOSEK      - objective sense       : min
MOSEK      - problem type          : LO (linear optimization problem)
MOSEK      - constraints           : 7                 variables              : 9
MOSEK      - integer variables     : 0
Optimizer started.
Simplex optimizer started.
```

```
Presolve started.
Presolve   - Stk. size (kb) : 0
Eliminator - tries                  : 0                   time                   : 0.00
Eliminator - elim's                 : 0
Lin. dep.  - tries                  : 0                   time                   : 0.00
Lin. dep.  - number                 : 0
Presolve terminated. Time: 0.00
Primal simplex optimizer started.
Primal simplex optimizer setup started.
Primal simplex optimizer setup terminated.
Optimizer  - solved problem         : the primal
Optimizer  - constraints            : 7                   variables              : 9
Optimizer  - hotstart               : yes
Optimizer  - Num. basic             : 7                   Basis rank             : 7
Optimizer  - Valid bas. fac.        : no
ITER      DEGITER(%)  PFEAS       DFEAS       POBJ                  DOBJ                  TIME      TOTTIME
0         0.00        0.00e+00    NA          1.4855737705e+01      NA                    0.00      0.01
0         0.00        0.00e+00    NA          1.4855737705e+01      NA                    0.00      0.01
Primal simplex optimizer terminated.
Simplex optimizer terminated. Time: 0.00.
Optimizer terminated. Time: 0.01
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.

5.9 Conic constraints

AMPL does not support conic constraints. It is of course possible to specify the conic quadratic constraint

$$x \geq \|z\|$$

in AMPL but it will be treated as a general nonlinear constraint. MOSEK cannot analyze the nonlinear constraints and discover that such a constraint is actually quadratic cone constraint. Therefore, an explicit method for specifying a conic constraints are needed.

The AMPL example

```
suffix cone integer; # Mosek specific suffix
option presolve 0;   # AMPL presolve should be turned off when the problem
                     # has conic constraints

var x;
var y;
```

```
var z;

minimize obj: y;
subject to con: x + z = y;

# Represents the conic constraint y >= sqrt(x^2+z^2)
let y.cone := -1;
let x.cone := 1;
let z.cone := 1;

option solver mosek;
option mosek_options 'outlev=2';
solve;
```

solves the example

$$\begin{array}{lccc} \text{minimize} & y & & \\ \text{subject to} & x+z & = & y, \\ & y \geq \sqrt{x^2+z^2}. & & \end{array} \tag{5.1}$$

The idea of the MOSEK specific extension is to use a variable suffix named `cone` to specify the index of the cone that variable is member of. If a variable is not a member of a cone, then `cone` suffix of the variable should not be specified. Alternatively the cone suffix can be set to 0. The `cone` suffix should be negative if the variable is on the left side of

$$x \geq \|z\|.$$

For each cone there should be at least one variable having a negative cone suffix and at most two variable can have negative cone suffix. If two variables have negative cone suffix, then a rotated quadratic cone is specified. Hence,

```
let y.cone := -1;
let x.cone := -1;
let z.cone := 1;
```

is the same as the rotated quadratic cone constraint

$$2xy \geq z^2 \text{ and } x, y \geq 0.$$

Finally, some observations about the MOSEK specific conic extension to AMPL are:

- A cone can contain as many variables as desired.
- A problem can contain any mixture of quadratic and rotated quadratic cones.
- The problem can contain any number cones.
- Currently, dual variables associated with constraints is not available in AMPL.
- Is is important that presolve is turned off when a problem has conic constraints because AMPL does not take such constraints into account. Hence, leaving the presolve on may lead to incorrect results.

5.10 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.

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

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

The resulting output is:

```
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


suffix up OUT;
suffix down OUT;
suffix current OUT;
Objective = Total_Cost
:  _var.down _var.current       _var.up        _var      :=
1   1.37385       1.84            1.86075    4.38525
2   1.8677        2.19      Infinity         0
3   1.82085       1.84      Infinity         0
4   1.35466       1.44      Infinity         0
5   1.57633       2.29      Infinity         0
```

```
6   0.094        0.77           0.794851   6.14754
7   1.22759      1.29      Infinity        0
8   0.57559      0.6            0.910769   3.42213
9   0.657279     0.72      Infinity        0
;
ampl: display _con.down,_con.current,_con.up,_con;
:     _con.down    _con.current   _con.up     _con       :=
1   -Infinity          2000      3965.37    0
2         297.6         350       375       0.0277049
3   -Infinity            55       172.029   0
4          63.0531      100       195.388   0.0267541
5   -Infinity           100       132.213   0
6   -Infinity           100       234.221   0
7          17.6923      100       142.821   0.0248361
;
```

5.11 Using the command line version of the AMPL interface

AMPL can generate a data file containing all the optimization problem and all relevant information which can then be read and solved by the MOSEK command line tool.

When the problem has been loaded into AMPL, the commands

```
ampl: option auxfiles rc;
ampl: write  bprob;
```

will make AMPL write the appropriate data files, i.e.

```
prob.nl
prob.col
prob.row
```

Then the problem can be solved using the command line version of MOSEK as follows

```
mosek prob.nl outlev=10 -a
```

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
```

For linear, quadratic and quadratic constrained problems a text file representation of the problem can be obtained using one of the commands

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

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

Interfaces 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\6\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

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

where

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

- $l^x \in \mathbb{R}^n$ is the lower limit on the activity for the variables.
- $u^x \in \mathbb{R}^n$ is the upper limit on the activity for the variables.

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

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

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. E.g.

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

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 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 so 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 certificate of primal infeasibility: The dual solution reported is a certificate of infeasibility, and the primal solution is undefined.

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

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

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 which 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 certificate of dual 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

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

where

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

and

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

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

We note that the dual of (9.4) is a problem which 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 Quadratic and quadratically constrained optimization

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

$$
\begin{array}{lccccll}
\text{minimize} & & & \frac{1}{2} x^T Q^o x + c^T x + c^f & & & \\
\text{subject to} & l_k^c & \leq & \frac{1}{2} x^T Q^k x + \sum_{j=0}^{n-1} a_{k,i} x_j & \leq & u_k^c, & k = 0, \ldots, m-1, \\
& l^x & \leq & x & \leq & u^x, & j = 0, \ldots, n-1,
\end{array}
\tag{9.5}
$$

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.

9.2.1 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 9.3.3.1 for further details.

9.2.2 Reformulating as a separable quadratic problem

The simplest quadratic optimization problem is

$$\begin{array}{lccl} \text{minimize} & 1/2x^TQx + c^Tx & & \\ \text{subject to} & Ax & = & b, \\ & x \geq 0. & & \end{array} \tag{9.6}$$

The problem (9.6) is said to be a separable problem if Q is a diagonal matrix or, in other words, if 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

$$Q = F^TF. \tag{9.7}$$

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

Using (9.7), the problem (9.6) can be reformulated as

$$\begin{array}{lccl} \text{minimize} & 1/2y^TIy + c^Tx & & \\ \text{subject to} & Ax & = & b, \\ & Fx - y & = & 0, \\ & x \geq 0. & & \end{array} \tag{9.8}$$

The problem (9.8) is also a quadratic optimization problem and has more constraints and variables than (9.6). 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 a dense matrix.

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

$$1/2x^T(F^TF)x \leq b \tag{9.9}$$

where F is a matrix and b is a scalar. (9.9) can be reformulated as

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

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

Next, consider the constraint

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

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, e.g. that D is a diagonal or a block diagonal matrix. If this is the case, it may be worthwhile performing the reformulation

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

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.

9.3 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

$$\begin{array}{lccccc} \text{minimize} & & & c^Tx + c^f & & \\ \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \\ & & & x \in \mathcal{C}. & & \end{array} \tag{9.10}$$

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 $x_t \in \mathbb{R}^{n_t}$. Please note that the n-dimensional Euclidean space $\mathbb{R}^n$ 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

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

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 9.3.3.

9.3.1 Duality for conic optimization

The dual problem corresponding to the conic optimization problem (9.10) is given by

$$\begin{array}{lccc}
\text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c & & \\
 & +(l^x)^T s_l^x - (u^x)^T s_u^x + c^f & & \\
\text{subject to} & A^T y + s_l^x - s_u^x + s_n^x & = & c, \\
 & -y + s_l^c - s_u^c & = & 0, \\
 & s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0, \\
 & s_n^x \in \mathcal{C}^* & &
\end{array} \tag{9.11}$$

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

- $\mathbb{R}$ set:
$$\mathcal{C}_t = \left\{ x \in \mathbb{R}^{n^t} \right\} \quad \Leftrightarrow \quad \mathcal{C}_t^* := \left\{ s \in \mathbb{R}^{n^t} : \; s = 0 \right\}.$$
- Quadratic cone:
$$\mathcal{C}_t := \left\{ x \in \mathbb{R}^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\} \quad \Leftrightarrow \quad \mathcal{C}_t^* = \mathcal{C}_t.$$

- Rotated quadratic cone:

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

Please note that the dual problem of the dual problem is identical to the original primal problem.

9.3.2 Infeasibility

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

9.3.3 Examples

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

9.3.3.1 Quadratic objective and constraints

From Section 9.2.2 we know that any convex quadratic problem can be stated on the form

$$\begin{array}{lccc} \text{minimize} & 0.5 \left\| Fx \right\|^2 + c^T x, & & \\ \text{subject to} & 0.5 \left\| Gx \right\|^2 + a^T x & \leq & b, \end{array} \tag{9.12}$$

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 (9.12) can be reformulated as

$$\begin{array}{lccc} \text{minimize} & 0.5 \left\| t \right\|^2 + c^T x, & & \\ \text{subject to} & 0.5 \left\| z \right\|^2 + a^T x & \leq & b, \\ & Fx - t & = & 0, \\ & Gx - z & = & 0 \end{array} \tag{9.13}$$

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.

$$\begin{array}{lcccl} \text{minimize} & v + c^T x, & & & \\ \text{subject to} & p + a^T x & = & b, & \\ & Fx - t & = & 0, & \\ & Gx - z & = & 0, & \\ & w & = & 1, & \\ & q & = & 1, & \\ & \left\| t \right\|^2 & \leq & 2vw, & v, w \geq 0, \\ & \left\| z \right\|^2 & \leq & 2pq, & p, q \geq 0. \end{array} \tag{9.14}$$

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

If we assume that F is a non-singular matrix — e.g. a diagonal matrix — then

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

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

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

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

9.3.3.2 Minimizing a sum of norms

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

$$\begin{array}{llll} \text{minimize} & \sum\limits_{i=1}^{k} \|x^i\| & & \\ \text{subject to} & Ax & = & b, \end{array} \tag{9.16}$$

where

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

This problem is equivalent to

$$\begin{array}{lllll} \text{minimize} & \sum\limits_{i=1}^{k} z_i & & & \\ \text{subject to} & Ax & = & b, & \\ & \|x^i\| & \leq & z_i, & i = 1, \ldots, k, \end{array} \tag{9.17}$$

which in turn is equivalent to

$$\begin{array}{llll} \text{minimize} & \sum\limits_{i=1}^{k} z_i & & \\ \text{subject to} & Ax & = & b, \\ & (z_i, x^i) \in \mathcal{C}_i, & & i = 1, \ldots, k \end{array} \tag{9.18}$$

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

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

The dual problem corresponding to (9.18) is

$$\begin{array}{lrcll} \text{maximize} & b^T y & & & \\ \text{subject to} & A^T y + s & = & c, & \\ & t_i & = & 1, & i = 1, \ldots, k, \\ & (t_i, s^i) \in \mathcal{C}_i, & & & i = 1, \ldots, k \end{array} \tag{9.19}$$

where

$$s := \left[\begin{array}{c} s^1 \\ \vdots \\ s^k \end{array} \right].$$

This problem is equivalent to

$$\begin{array}{lrcll} \text{maximize} & b^T y & & & \\ \text{subject to} & A^T y + s & = & c, & \\ & \left\| s^i \right\|_2^2 & \leq & 1, & i = 1, \ldots, k. \end{array} \tag{9.20}$$

Please note that in this case the dual problem can be reduced to an "ordinary" convex quadratically constrained optimization problem 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 (9.19) rather than the primal problem (9.18).

9.3.3.3 Modelling polynomial terms using conic optimization

Generally an arbitrary polynomial term of the form

$$f x^g$$

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.

E.g., consider the problem

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

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

$$\begin{array}{lllll} \text{minimize} & c^T x & & & \\ \text{subject to} & \sum_{j=1}^{n} f_j z_j & = & b, & \\ & v_j & = & \sqrt{2}, & j = 1, \ldots, n, \\ & v_j^2 & \leq & 2 z_j x_j, & j = 1, \ldots, n, \\ & x, z \geq 0, & & & \end{array} \tag{9.21}$$

because

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

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 (9.21) 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

$$\begin{array}{rcl} t^2 & \leq & 2xz, \\ z & = & 0.5, \\ x, z, & \geq & 0. \end{array} \tag{9.22}$$

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

$$\begin{array}{rcl} x^{1.5} & \leq & t, \\ 0 & \leq & x. \end{array} \tag{9.23}$$

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

$$\begin{array}{rcl} x^2 & \leq & 2st, \\ s, t & \geq & 0. \end{array} \tag{9.24}$$

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

$$\begin{array}{rcl} x^2 & \leq & 2st, \\ w^2 & \leq & 2vr, \\ x & = & v, \\ s & = & w, \\ r & = & \frac{1}{8}, \\ s,t,v,r & \geq & 0, \end{array} \tag{9.25}$$

then

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

Moreover,

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

leading to the conclusion that

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

(9.25) is a conic reformulation which is equivalent to (9.23). Please note that the $x \geq 0$ constraint does not appear explicitly in (9.24) and (9.25), but implicitly since $x = v \geq 0$.

As we shall see next, any polynomial term of the form x^g where g is a positive rational number can be represented using conic quadratic constraints [2, pp. 12-13], [11].

9.3.3.4 Optimization with rational polynomials

We next demonstrate how to model convex polynomial constraints of the form $x^{p/q} \leq t$ (where p and q are both positive integers) as a set of rotated quadratic cone constraints.

Following Ben-Tal et al. [11, p. 105] we use an intermediate result, namely that the set

$$\{s \in \mathbb{R},\, y \in \mathbb{R}_+^{2^l} \mid s \leq (2^{l2^{l-1}} y_1 y_2 \cdots y_{2^l})^{1/2^l}\}$$

is convex and can be represented as a set of rotated quadratic cone constraints. To see this, we rewrite the condition (exemplified for $l = 3$),

$$s \leq \left(2^{12} \cdot y_1 \cdot y_2 \cdot y_3 \cdot y_4 \cdot y_5 \cdot y_6 \cdot y_7 \cdot y_8\right)^{1/8} \tag{9.26}$$

as

$$s^8 \leq \left(2^{12} \cdot y_1 \cdot y_2 \cdot y_3 \cdot y_4 \cdot y_5 \cdot y_6 \cdot y_7 \cdot y_8\right) \tag{9.27}$$

since all $y_i \geq 0$. We next introduce l levels of auxiliary variables and (rotated cone) constraints

$$y_{11}^2 \leq 2y_1y_2, \quad y_{12}^2 \leq 2y_3y_4, \quad y_{13}^2 \leq 2y_5y_6, \quad y_{14}^2 \leq 2y_7y_8, \tag{9.28}$$

$$y_{21}^2 \leq 2y_{11}y_{12}, \quad y_{22}^2 \leq 2y_{13}y_{14}, \tag{9.29}$$

and finally

$$s^2 \leq 2y_{21}y_{22}. \tag{9.30}$$

By simple substitution we see that (9.30) and (9.27) are equivalent, and since (9.30) involves only a set of simple rotated conic constraints then the original constraint (9.26) can be represented using only rotated conic constraints.

9.3.3.5 Convex increasing power functions

Using the intermediate result in section 9.3.3.4 we can include convex power functions with positive rational powers, i.e., constraints of the form

$$x^{p/q} \le t, \quad x \ge 0$$

where p and q are positive integers and $p/q \ge 1$. For example, consider the constraints

$$x^{5/3} \le t, \quad x \ge 0.$$

We rewrite it as

$$x^8 \le x^3 t^3, \quad x \ge 0$$

which in turn is equivalent to

$$x^8 \le 2^{12} y_1 y_2 \cdots y_8, \quad x = y_1 = y_2 = y_3, \quad y_4 = y_5 = y_6 = t, \quad y_6 = 1, \quad y_7 = 2^{-12}, \quad x, y_i \ge 0,$$

i.e., it can be represented as a set of rotated conic and linear constraints using the reformulation above.

For general p and q we choose l as the smallest integer such that $p \le 2^l$ and we construct the problem as

$$x^{2^l} \le 2^{l2^{l-1}} y_1 y_2 \cdots y_{2^l}, \quad x, y_i \ge 0,$$

with the first $2^l - p$ elements of y set to x, the next q elements set to t, and the product of the remaining elements as $1/2^{l2^{l-1}}$, i.e.,

$$x^{2^l} \le x^{2^l - p} t^q, \quad x \ge 0 \qquad \Longleftrightarrow \qquad x^{p/q} \le t, \quad x \ge 0.$$

9.3.3.6 Decreasing power functions

We can also include decreasing power functions with positive rational powers

$$x^{-p/q} \le t, \quad x \ge 0$$

where p and q are positive integers. For example, consider

$$x^{-5/2} \le t, \quad x \ge 0,$$

or equivalently

$$1 \le x^5 t^2, \quad x \ge 0,$$

which, in turn, can be rewritten as

$$s^8 \le 2^{12} y_1 y_2 \cdots y_8, \quad s = 2^{3/2}, \quad y_1 = \cdots = y_5 = x, \quad y_6 = y_7 = y_8 = t, \quad x, y_i \ge 0.$$

For general p and q we choose l as the smallest integer such that $p + q \le 2^l$ and we construct the problem as

$$s^{2^l} \le y_1 y_2 \cdots y_{2^l}, \quad y_i \ge 0,$$

with $s = 2^{l/2}$ and the first p elements of y set to x, the next q elements set to t, and the remaining elements set to 1, i.e.,

$$1 \le x^p t^q, \quad x \ge 0 \qquad \Longleftrightarrow \qquad x^{-p/q} \le t, \quad x \ge 0.$$

9.3.3.7 Minimizing general polynomials

Using the formulations in section 9.3.3.5 and section 9.3.3.6 it is straightforward to minimize general polynomials. For example, we can minimize

$$f(x) = x^2 + x^{-2}$$

which is used in statistical matching. We first formulate the problem

$$\begin{array}{ll} \text{minimize} & u + v \\ \text{subject to} & x^2 \leq u \\ & x^{-2} \leq v, \end{array}$$

which is equivalent to the quadratic conic optimization problem

$$\begin{array}{ll} \text{minimize} & u + v \\ \text{subject to} & x^2 \leq 2uw \\ & s^2 \leq 2y_{21}y_{22} \\ & y_{21}^2 \leq 2y_1y_2 \\ & y_{22}^2 \leq 2y_3y_4 \\ & w = 1 \\ & s = 2^{3/4} \\ & y_1 = y_2 = x \\ & y_3 = v \\ & y_4 = 1 \end{array}$$

in the variables $(x, u, v, w, s, y_1, y_2, y_3, y_4, y_{21}, y_{22})$.

9.3.3.8 Further reading

If you want to learn more about what can be modeled as a conic optimization problem we recommend the references [2, 11, 16].

9.3.4 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.

9.3.4.1 Non-attainment in the primal problem

Consider the example

$$\begin{array}{lrcl} \text{minimize} & z & & \\ \text{subject to} & 2yz & \geq & x^2, \\ & x & = & \sqrt{2}, \\ & y, z & \geq & 0, \end{array} \tag{9.31}$$

which corresponds to the problem

$$\begin{array}{lrcl} \text{minimize} & \frac{1}{y} & & \\ \text{subject to} & y & \geq & 0. \end{array} \tag{9.32}$$

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

9.3.4.2 Non-attainment in the dual problem

Next, consider the example

$$\begin{array}{lrcl} \text{minimize} & x_4 & & \\ \text{subject to} & x_3 + x_4 & = & 1, \\ & x_1 & = & 0, \\ & x_2 & = & 1, \\ & 2x_1x_2 & \geq & x_3^2, \\ & x_1, x_2 & \geq & 0, \end{array} \tag{9.33}$$

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 (9.33) is

$$\begin{array}{lrcl} \text{maximize} & y_1 + y_3 & & \\ \text{subject to} & y_2 + s_1 & = & 0, \\ & y_3 + s_2 & = & 0, \\ & y_1 + s_3 & = & 0, \\ & y_1 & = & 1, \\ & 2s_1s_2 & \geq & s_3^2, \\ & s_1, s_2 & \geq & 0. \end{array} \tag{9.34}$$

Therefore,

$$y_1^* = 1$$

and

$$s_3^* = -1.$$

This implies that

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

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, no primal-dual bounded optimal solution with zero duality gap exists. Of course it is possible to find a primal-dual feasible solution such that the duality gap is close to zero, but then s_1^* will be similarly large. This is likely to make the problem (9.33) hard to solve.

An inspection of the problem (9.33) 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 (9.33) or eliminate x_1 and x_3 from the problem it becomes easy to solve.

9.4 Nonlinear convex optimization

MOSEK is capable of solving smooth (twice differentiable) convex nonlinear optimization problems of the form

$$\begin{array}{lccccl} \text{minimize} & & & f(x) + c^T x & & \\ \text{subject to} & & & g(x) + Ax - x^c & = & 0, \\ & l^c & \leq & x^c & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \end{array} \tag{9.35}$$

where

- m is the number of constraints.
- n is the number of decision variables.
- $x \in \mathbb{R}^n$ is a vector of decision variables.
- $x^c \in \mathbb{R}^m$ is a vector of constraints or slack variables.
- $c \in \mathbb{R}^n$ is the linear part objective function.
- $A \in \mathbb{R}^{m \times n}$ is the constraint matrix.
- $l^c \in \mathbb{R}^m$ is the lower limit on the activity for the constraints.
- $u^c \in \mathbb{R}^m$ is the upper limit on the activity for the constraints.
- $l^x \in \mathbb{R}^n$ is the lower limit on the activity for the variables.
- $u^x \in \mathbb{R}^n$ is the upper limit on the activity for the variables.
- $f : \mathbb{R}^n \to \mathbb{R}$ is a nonlinear function.
- $g : \mathbb{R}^n \to \mathbb{R}^m$ 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 in all $x \in [l^x; u^x]$. Moreover, $f(x)$ must be a convex function and $g_i(x)$ must satisfy

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

9.4.1 Duality

So far, we have not discussed what happens when MOSEK is used to solve a primal or dual infeasible problem. In the following 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

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

which is equivalent to

$$\begin{array}{lccc} \text{maximize} & f(x) - y^T g(x) - x^T(\nabla f(x)^T - \nabla g(x)^T y) & & \\ & +((l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f & & \\ \text{subject to} & -\nabla f(x)^T + A^T y + \nabla g(x)^T y + s_l^x - s_u^x & = & c, \\ & -y + s_l^c - s_u^c & = & 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0. & & \end{array} \tag{9.36}$$

9.5 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 and therefore causing 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 either are 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 may be a good idea to introduce more constraints and variables if it makes the model separable. Furthermore, a big but sparse problem may 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. [3, 4, 14, 15].

9.5.1 Avoid near infeasible models

Consider the linear optimization problem

$$\begin{array}{lrcl} \text{minimize} & & & \\ \text{subject to} & x + y & \leq & 10^{-10} + \alpha, \\ & 1.0e4x + 2.0e4y & \geq & 10^{-6}, \\ & x, y \geq 0. & & \end{array} \tag{9.37}$$

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.

9.6 Examples continued

9.6.1 The absolute value

Assume that we have a constraint for the form

$$|f^T x + g| \leq b \tag{9.38}$$

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

It is easy to verify that the constraint (9.38) is equivalent to

$$-b \leq f^T x + g \leq b \tag{9.39}$$

which is a set of ordinary linear inequality constraints.

Please note that equalities involving an absolute value such as

$$|x| = 1$$

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

9.6.2 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 a 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

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

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.

9.6.2.1 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

$$\begin{array}{lccc} \text{minimize} & V[r^T(w^0+x)] & & \\ \text{subject to} & E[r^T(w^0+x)] & = & t, \\ & e^T x & = & 0, \end{array} \tag{9.40}$$

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

$$\begin{array}{lccc} \text{minimize} & (w^0+x)^T\Sigma(w^0+x) & & \\ \text{subject to} & \bar{r}^T(w^0+x) & = & t, \\ & e^T x & = & 0. \end{array} \tag{9.41}$$

9.6.2.2 Conic quadratic reformulation

An equivalent conic quadratic reformulation is given by:

$$\begin{array}{lccc} \text{minimize} & f & & \\ \text{subject to} & \Sigma^{\frac{1}{2}}(w^0+x)-g & = & 0, \\ & \bar{r}^T(w^0+x) & = & t, \\ & e^T x & = & 0, \\ & f \geq \|g\|. & & \end{array} \tag{9.42}$$

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.

9.6.2.3 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 modeled as follows

$$\text{commission} + \frac{\text{bid}}{\text{ask}} - \text{spread} + \theta\sqrt{\frac{\text{trade volume}}{\text{daily volume}}}, \tag{9.43}$$

and that it is important to incorporate these 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:

$$\begin{array}{lccl}\text{minimize} & f & & \\ \text{subject to} & \Sigma^{\frac{1}{2}}(w^0+x)-g & = & 0, \\ & \bar{r}^T(w^0+x) & = & t, \\ & e^Tx+e^Ty & = & 0, \\ & |x_j|(m_j|x_j|^{\frac{1}{2}}) & \leq & y_j, \\ & f \geq \|g\|. & & \end{array} \tag{9.44}$$

Now, defining the variable transformation

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

we obtain

$$\begin{array}{lccl}\text{minimize} & f & & \\ \text{subject to} & \Sigma^{\frac{1}{2}}(w^0+x)-g & = & 0, \\ & \bar{r}^T(w^0+x) & = & t, \\ & e^Tx+m^T\bar{y} & = & 0, \\ & |x_j|^{3/2} & \leq & \bar{y}_j, \\ & f \geq \|g\|. & & \end{array} \tag{9.45}$$

As shown in Section 9.3.3.3 the set

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

can be modeled by

$$\begin{array}{rcl} x_j & \leq & z_j, \\ -x_j & \leq & z_j, \\ z_j^2 & \leq & 2s_j\bar{y}_j, \\ u_j^2 & \leq & 2v_jq_j, \\ z_j & = & v_j, \\ s_j & = & u_j, \\ q_j & = & \frac{1}{8}, \\ q_j,s_j,\bar{y}_j,v_j,q_j & \geq & 0. \end{array} \tag{9.46}$$

9.6.2.4 Further reading

For further reading please see [17] in particular, and [20] 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: Solve the problem using selected method.

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 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 [3, 4].

It is possible to fine-tune the behavior of the presolve or to turn it off entirely. If presolve consumes too much time or memory compared to the reduction in problem size gained it may be disabled. 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

- the eliminator, and
- the linear dependency check.

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

10.1.1.1 Eliminator

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.

If the eliminator consumes too much time or memory compared to the reduction in problem size gained it may be disabled. This can be done with the parameter MSK_IPAR_PRESOLVE_ELIMINATOR_USE to MSK_OFF.

10.1.1.2 Linear dependency checker

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. 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.

It is best practise to build models without linear dependencies. If the linear dependencies are removed at the modeling 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

All linear, conic, and convex optimization problems have an equivalent dual problem associated with them. MOSEK has built-in heuristics to determine if it is most efficient to solve the primal or dual problem. The form (primal or dual) solved is displayed in the MOSEK log. Should the internal heuristics not choose the most efficient form of the problem it may be worthwhile to set the dualizer 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.

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 may 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.

10.2 Linear optimization

10.2.1 Optimizer selection

Two different types of optimizers are available for linear problems: The default is an interior-point method, and the alternatives are simplex methods. The optimizer can be selected using the parameter MSK_IPAR_OPTIMIZER.

10.2.2 The interior-point optimizer

The purpose of this section is to provide information about the algorithm employed in MOSEK interior-point optimizer.

In order to keep the discussion simple it is assumed that MOSEK solves linear optimization problems on standard form

$$\begin{array}{lccc} \text{minimize} & c^T x & & \\ \text{subject to} & Ax & = & b, \\ & x \geq 0. & & \end{array} \tag{10.1}$$

This is in fact what happens inside MOSEK; for efficiency reasons MOSEK converts the problem to standard form before solving, then convert it back to the input form when reporting the solution.

Since it is not known beforehand whether problem (10.1) has an optimal solution, is primal infeasible or is dual infeasible, the optimization algorithm must deal with all three situations. This is the reason that MOSEK solves the so-called homogeneous model

$$\begin{array}{rcl} Ax - b\tau & = & 0, \\ A^T y + s - c\tau & = & 0, \\ -c^T x + b^T y - \kappa & = & 0, \\ x, s, \tau, \kappa & \geq & 0, \end{array} \tag{10.2}$$

where y and s correspond to the dual variables in (10.1), and τ and κ are two additional scalar variables. Note that the homogeneous model (10.2) always has solution since

$$(x, y, s, \tau, \kappa) = (0, 0, 0, 0, 0)$$

is a solution, although not a very interesting one.

Any solution

$$(x^*, y^*, s^*, \tau^*, \kappa^*)$$

to the homogeneous model (10.2) satisfies

$$x_j^* s_j^* = 0 \text{ and } \tau^* \kappa^* = 0.$$

Moreover, there is always a solution that has the property

$$\tau^* + \kappa^* > 0.$$

First, assume that $\tau^* > 0$. It follows that

$$\begin{array}{rcl} A \frac{x^*}{\tau^*} & = & b, \\ A^T \frac{y^*}{\tau^*} + \frac{s^*}{\tau^*} & = & c, \\ -c^T \frac{x^*}{\tau^*} + b^T \frac{y^*}{\tau^*} & = & 0, \\ x^*, s^*, \tau^*, \kappa^* & \geq & 0. \end{array} \tag{10.3}$$

This shows that $\frac{x^*}{\tau^*}$ is a primal optimal solution and $(\frac{y^*}{\tau^*}, \frac{s^*}{\tau*})$ is a dual optimal solution; this is reported as the optimal interior-point solution since

$$(x, y, s) = \left(\frac{x^*}{\tau^*}, \frac{y^*}{\tau^*}, \frac{s^*}{\tau*}\right)$$

is a primal-dual optimal solution.

On other hand, if $\kappa^* > 0$ then

$$\begin{array}{rcl} Ax^* & = & 0, \\ A^T y^* + s^* & = & 0, \\ -c^T x^* + b^T y^* & = & \kappa^*, \\ x^*, s^*, \tau^*, \kappa^* & \geq & 0. \end{array} \tag{10.4}$$

This implies that at least one of

$$-c^T x^* > 0 \tag{10.5}$$

or

$$b^T y^* > 0 \tag{10.6}$$

is satisfied. If (10.5) is satisfied then x^* is a certificate of dual infeasibility, whereas if (10.6) is satisfied then y^* is a certificate of dual infeasibility.

In summary, by computing an appropriate solution to the homogeneous model, all information required for a solution to the original problem is obtained. A solution to the homogeneous model can be computed using a primal-dual interior-point algorithm [9].

10.2.2.1 Interior-point termination criterion

For efficiency reasons it is not practical to solve the homogeneous model exactly. Hence, an exact optimal solution or an exact infeasibility certificate cannot be computed and a reasonable termination criterion has to be employed.

In every iteration, k, of the interior-point algorithm a trial solution

$$(x^k, y^k, s^k, \tau^k, \kappa^k)$$

to homogeneous model is generated where

$$x^k, s^k, \tau^k, \kappa^k > 0.$$

Whenever the trial solution satisfies the criterion

$$\begin{array}{rcl} \left\| A\frac{x^k}{\tau^k} - b \right\| & \leq & \varepsilon_p (1 + \|b\|), \\ \left\| A^T \frac{y^k}{\tau^k} + \frac{s^k}{\tau^k} - c \right\| & \leq & \varepsilon_d (1 + \|c\|), \text{ and} \\ \min\left(\frac{(x^k)^T s^k + \tau^k \kappa^k}{(\tau^k)^2}, \left| \frac{c^T x^k}{\tau^k} - \frac{b^T y^k}{\tau^k} \right| \right) & \leq & \varepsilon_g \max\left(1, \left| \frac{c^T x^k}{\tau^k} \right| \right), \end{array} \tag{10.7}$$

the interior-point optimizer is terminated and

$$\frac{(x^k, y^k, s^k)}{\tau^k}$$

is reported as the primal-dual optimal solution. The interpretation of (10.7) is that the optimizer is terminated if

Tolerance	Parameter name
ε_p	MSK_DPAR_INTPNT_TOL_PFEAS
ε_d	MSK_DPAR_INTPNT_TOL_DFEAS
ε_g	MSK_DPAR_INTPNT_TOL_REL_GAP
ε_i	MSK_DPAR_INTPNT_TOL_INFEAS

Table 10.1: Parameters employed in termination criterion.

- $\frac{x^k}{\tau^k}$ is approximately primal feasible,
- $\left(\frac{y^k}{\tau^k}, \frac{s^k}{\tau^k}\right)$ is approximately dual feasible, and
- the duality gap is almost zero.

On the other hand, if the trial solution satisfies

$$-\varepsilon_i c^T x^k > \frac{\|c\|}{\max(\|b\|, 1)} \left\|Ax^k\right\| \tag{10.8}$$

then the problem is declared dual infeasible and x^k is reported as a certificate of dual infeasibility. The motivation for this stopping criterion is as follows: First assume that $\left\|Ax^k\right\| = 0$; then x^k is an exact certificate of dual infeasibility. Next assume that this is not the case, i.e.

$$\left\|Ax^k\right\| > 0,$$

and define

$$\bar{x} := \varepsilon_i \frac{\max(1, \|b\|) x^k}{\|Ax^k\| \, \|c\|}.$$

It is easy to verify that

$$\|A\bar{x}\| = \varepsilon_i \text{ and } -c^T \bar{x} > 1,$$

which shows $\bar{x}$ is an approximate certificate dual infeasibility where ε_i controls the quality of the approximation. A smaller value means a better approximation.

Finally, if

$$\varepsilon_i b^T y^k \geq \frac{\|b\|}{\max(1, \|c\|)} \left\|A^T y^k + s^k\right\| \tag{10.9}$$

then y^k is reported as a certificate of primal infeasibility.

It is possible to adjust the tolerances ε_p, ε_d, ε_g and ε_i using parameters; see table 10.1 for details.

The default values of the termination tolerances are chosen such that for a majority of problems appearing in practice it is not possible to achieve much better accuracy. Therefore, tightening the tolerances usually is not worthwhile. However, an inspection of (10.7) reveals that quality of the solution is dependent on $\|b\|$ and $\|c\|$; the smaller the norms are, the better the solution accuracy.

The interior-point method as implemented by MOSEK will converge toward optimality and primal and dual feasibility at the same rate [9]. This means that if the optimizer is stopped prematurely then it is very unlikely that either the primal or dual solution is feasible. Another consequence is that in most cases all the tolerances, ε_p, ε_d and ε_g, has to be relaxed together to achieve an effect.

The basis identification discussed in section 10.2.2.2 requires an optimal solution to work well; hence basis identification should turned off if the termination criterion is relaxed.

To conclude the discussion in this section, relaxing the termination criterion is usually is not worthwhile.

10.2.2.2 Basis identification

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 may be found in [6].

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.

10.2.2.3 The interior-point log

Below is a typical log output from the interior-point optimizer presented:

```
Optimizer  - threads                : 1
Optimizer  - solved problem         : the dual
Optimizer  - constraints            : 2                 variables              : 6
Factor     - setup time             : 0.04              order time             : 0.00
Factor     - GP order used          : no                GP order time          : 0.00
Factor     - nonzeros before factor : 3                 after factor           : 3
Factor     - offending columns      : 0                 flops                  : 1.70e+001
ITE PFEAS    DFEAS    KAP/TAU  POBJ               DOBJ               MU       TIME
0   2.0e+002 2.9e+001 2.0e+002 -0.000000000e+000 -1.204741644e+003 2.0e+002 0.44
1   2.2e+001 3.1e+000 7.3e+002 -5.885951891e+003 -5.856764353e+003 2.2e+001 0.57
2   3.8e+000 5.4e-001 9.7e+001 -7.405187479e+003 -7.413054916e+003 3.8e+000 0.58
3   4.0e-002 5.7e-003 2.6e-001 -7.664507945e+003 -7.665313396e+003 4.0e-002 0.58
4   4.2e-006 6.0e-007 2.7e-005 -7.667999629e+003 -7.667999714e+003 4.2e-006 0.59
5   4.2e-010 6.0e-011 2.7e-009 -7.667999994e+003 -7.667999994e+003 4.2e-010 0.59
```

The first line displays the number of threads used by the optimizer and second line tells that the optimizer choose to solve the dual problem rather the primal problem. The next line displays the problem dimensions as seen by the optimizer, and the "`Factor...`" lines show various statistics. This is followed by the iteration log.

Using the same notation as in section 10.2.2 the columns of the iteration log has the following meaning:

- `ITE`: Iteration index.

- PFEAS: $\left\|Ax^k - b\tau^k\right\|$. The numbers in this column should converge monotonically towards to zero.
- DFEAS: $\left\|A^T y^k + s^k - c\tau^k\right\|$. The numbers in this column should converge monotonically toward to zero.
- KAP/TAU: κ^k/τ^k. If the numbers in this column converge toward zero then the problem has an optimal solution. Otherwise if the numbers converge towards infinity, the problem is primal or/and dual infeasible.
- POBJ: $c^T x^k/\tau^k$. An estimate for the primal objective value.
- DOBJ: $b^T y^k/\tau^k$. An estimate for the dual objective value.
- MU: $\frac{(x^k)^T s^k + \tau^k \kappa^k}{n+1}$. The numbers in this column should always converge monotonically to zero.
- TIME: Time spend since the optimization started.

10.2.3 The simplex based optimizer

An alternative to the interior-point optimizer is the simplex optimizer.

The simplex optimizer uses a different method that allows exploiting an initial guess for the optimal solution to reduce the solution time. Depending on the problem it may be faster or slower to use an initial guess; 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 for the given 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

Though MOSEK is designed to minimize numerical instability, completely avoiding it is impossible when working in finite precision. MOSEK counts a "numerical unexpected behavior" event inside the optimizer as a *set-back*. The user can define how many set-backs the optimizer accepts; 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.

Set-backs are, for example, repeated singularities when factorizing the basis matrix, repeated loss of feasibility, degeneracy problems (no progress in objective) and 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. Then, if a lot of set-backs still occur, 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 impossible, however, in recent years the dual optimizer has seen 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

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

To use the network simplex optimizer, do the following:

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

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 in section 10.3.1 is applied, MOSEK will attemt to exploit this structure to speed up the optimization.

This is done heuristically by detecting the largest network embedded in the problem, solving this subproblem using the network simplex optimizer, and using the 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.

If MOSEK only finds limited network structure in a problem, consider trying to switch off presolve `MSK_IPAR_PRESOLVE_USE` and scaling `MSK_IPAR_SIM_SCALING`, since in rare cases it might disturb the network heuristic.

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.

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 [5].

10.4.1.1 Interior-point termination criteria

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

10.5 Nonlinear convex optimization

10.5.1 The interior-point optimizer

For quadratic, quadratically constrained, and general convex optimization problems 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 [7, 8].

10.5.1.1 The convexity requirement

Continuous nonlinear problems are required to be convex. For quadratic problems MOSEK test this requirement before optimizing. Specifying a non-convex problem results in an error message.

The following parameters are available to control the convexity check:

- MSK_IPAR_CHECK_CONVEXITY: Turn convexity check on/off.
- MSK_DPAR_CHECK_CONVEXITY_REL_TOL: Tolerance for convexity check.
- MSK_IPAR_LOG_CHECK_CONVEXITY: Turn on more log information for debugging.

10.5.1.2 The differentiabilty requirement

The nonlinear optimizer in MOSEK requires both first order and second order derivatives. This of course implies care should be taken when solving problems involving non-differentiable functions.

For instance, the function

$$f(x) = x^2$$

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.

is differentiable everywhere whereas the function

$$f(x) = \sqrt{x}$$

is only diffrentiable for $x > 0$. In order to make sure that MOSEK evaulates the functions at points where they are differentiable, the function domains must be defined by setting appropriate variable bounds.

In general, if a variable is not ranged MOSEK will only evaluate that variable at points strictly within the bounds. Hence, imposing the bound

$$x \geq 0$$

in the case of $\sqrt{x}$ is sufficient to guarantee that the function will only be evaluated in points where it is differentiable.

However, if a function is differentiable on closed a range, specifying the variable bounds is not sufficient. Consider the function

$$f(x) = \frac{1}{x} + \frac{1}{1-x}. \tag{10.10}$$

In this case the bounds

$$0 \leq x \leq 1$$

will not guarantee that MOSEK only evalues the function for x between 0 and 1. To force MOSEK to strictly satisfy both bounds on ranged variables set the parameter MSK_IPAR_INTPNT_STARTING_POINT to MSK_STARTING_POINT_SATISFY_BOUNDS.

For efficiency reasons it may be better to reformulate the problem than to force MOSEK to observe ranged bounds strictly. For instance, (10.10) can be reformulated as follows

$$\begin{array}{rcl} f(x) & = & \frac{1}{x} + \frac{1}{y} \\ 0 & = & 1 - x - y \\ 0 & \leq & x \\ 0 & \leq & y. \end{array}$$

10.5.1.3 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.

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

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.

```
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.

10.7 Understanding solution quality

MOSEK will, in general, not produce an *exact* optimal solution; for efficiency reasons computations are performed in finite precision. This means that it is important to evaluate the quality of the reported solution. To evaluate the solution quality inspect the following properties:

- The solution status reported by MOSEK.
- Primal feasibility: How much the solution violates the original constraints of the problem.
- Dual feasibility: How much the dual solution violates the constraints of the dual problem.
- Duality gap: The difference between the primal and dual objective values.

Ideally, the primal and dual solutions should only violate the constraints of their respective problem *slightly* and the primal and dual objective values should be *close*. This should be evaluated in the context of the problem: How good is the data precision in the problem, and how exact a solution is required.

10.7.1 The solution summary

The solution summary is a small display generated by MOSEK that makes it easy to check the quality of the solution.

10.7.1.1 The optimal case

The solution summary has the format

```
Problem status  : PRIMAL_AND_DUAL_FEASIBLE
Solution status : OPTIMAL
Primal - objective: 5.5018458883e+03   eq. infeas.: 1.20e-12 max bound infeas.: 2.31e-14
Dual   - objective: 5.5018458883e+03   eq. infeas.: 1.15e-14 max bound infeas.: 7.11e-15
```

i.e. it shows status information, objective values and quality measures for the primal and dual solutions.

Assumeing that we are solving a linear optimization problem and referring to the problems (9.1) and (9.2), the interpretation of the solution summary is as follows:

- Problem status: The status of the problem.
- Solution status: The status of the solution.
- Primal objective: The primal objective value.
- Primal eq. infeas: $\|Ax^x - x^c\|_\infty$.
- Primal max bound infeas.: $\max(l^c - x^c; x^c - u^c; l^x - x^x; x^x - u^x; 0)$.

- Dual objective: The dual objective value.
- Dual eq. infeas: $\left\| -y + s_l^c - s_u^c; A^T y + s_l^x - s_u^x - c \right\|_\infty$.
- Dual max bound infeas.: $\max(-s_l^c; -s_u^c; -s_l^x; -s_u^x; 0)$.

In the solution summary above the solution is classified as `OPTIMAL`, meaning that the solution should be a good approximation to the true optimal solution. This seems very reasonable since the primal and dual solutions only violate their respective constraints slightly. Moreover, the duality gap is small, i.e. the primal and dual objective values are almost identical.

10.7.1.2 The primal infeasible case

For an infeasible problem the solution summary might look like this:

```
Problem status  : PRIMAL_INFEASIBLE
Solution status : PRIMAL_INFEASIBLE_CER
Primal - objective: 0.0000000000e+00   eq. infeas.: 0.00e+00 max bound infeas.: 0.00e+00
Dual   - objective: 1.0000000000e+02   eq. infeas.: 0.00e+00 max bound infeas.: 0.00e+00
```

It is known that if the problem is primal infeasible then an infeasibility certificate exists, which is a solution to the problem (9.3) having a positive objective value. Note that the primal solution plays no role and only the dual solution is used to specify the certificate.

Therefore, in the primal infeasible case the solution summery should report how good the dual solution is to the problem (9.3). The interpretation of the solution summary is as follows:

- Problem status: The status of the problem.
- Solution status: The status of the solution.
- Primal objective: Should be ignored.
- Primal eq. infeas: Should be ignored.
- Primal max bound infeas.: Should be ignored.
- Dual objective: $(l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x$.
- Dual eq. infeas: $\left\| -y + s_l^c - s_u^c; A^T y + s_l^x - s_u^x - 0 \right\|_\infty$.
- Dual max bound infeas.: $\max(-s_l^c; -s_u^c; -s_l^x; -s_u^x)$.

Please note that

- any information about the primal solution should be ignored.
- the dual objective value should be strictly positive if primal problem is minimization problem. Otherwise it should be strictly negative.
- the bigger the ratio

$$\frac{(l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x}{\max(\| -y + s_l^c - s_u^c; A^T y + s_l^x - s_u^x - 0 \|_\infty, \max(-s_l^c; -s_u^c; -s_l^x; -s_u^x))}$$

  is, the better the certificate is. The reason is that a certificate is a ray, and hence only the direction is important. Therefore, in principle, the certificate should be normalized before using it.

Please see Section 12.2 for more information about certificates of infeasibility.

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 [23] by Wolsey is a good introduction to integer optimization.

11.1 Some notation

In general, an integer optimization problem has the form

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

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

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

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.
MSK_IPAR_MIO_MAX_NUM_SOLUTIONS	Yes	Maximum number of feasible integer solutions 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 [23].

11.6 Understanding solution quality

To determine the quality of the solution one should check the following:

- The solution status key returned by MOSEK.
- The *optimality gap*: A messure for how much the located solution can deviate from the optimal solution to the problem.
- Feasibility. How much the solution violates the constraints of the problem.

The *optimality gap* is a measure for how close the solution is to the optimal solution. The optimality gap is given by

$$\epsilon = |(\text{objective value of feasible solution}) - (\text{objective bound})|.$$

The objective value of the solution is guarentted to be within ϵ of the optimal solution.

The optimality gap can be retrived through the solution item `MSK_DINF_MIO_OBJ_ABS_GAP`. Often it is more meaningful to look at the optimality gap normalized with the magnitude of the solution. The relative optimality gap is available in `MSK_DINF_MIO_OBJ_REL_GAP`.

11.6.1 Solutionsummary

After a call to the optimizer the solution summary might look like this:

```
Problem status  : PRIMAL_FEASIBLE
Solution status : INTEGER_OPTIMAL
Primal - objective: 1.2015000000e+06   eq. infeas.: 0.00e+00 max bound infeas.: 0.00e+00
cone infeas.: 0.00e+00 integer infeas.: 0.00e+00
```

The second line contains the solution status key. This shows how MOSEK classified the solution. In this case it is `INTEGER_OPTIMAL`, meaning that the solution is considered to be optimal within the selected tolerances.

The third line contains information relating to the solution. The first number is the primal objective function. The second and third number is the maximum infeasibility in the equality constraints and bounds respectfully. The fourth and fifth number is the maximum infeasibility in the conic and integral contraints. All the numbers relating to the feasibility of the solution should be small for the solution to be valid.

Chapter 12

The analyzers

12.1 The problem analyzer

The problem analyzer prints a detailed survey of the model's

- linear constraints and objective
- quadratic constraints
- conic constraints
- variables

In the initial stages of model formulation the problem analyzer may be used as a quick way of verifying that the model has been built or imported correctly. In later stages it can help revealing special structures within the model that may be used to tune the optimizer's performance or to identify the causes of numerical difficulties.

The problem analyzer is run from the command line using the `-anapro` argument and produces something similar to the following (this is the problemanalyzer's survey of the `aflow30a` problem from the MIPLIB 2003 collection, see Appendix J for more examples):

```
Analyzing the problem

Constraints              Bounds                   Variables
 upper bd:       421      ranged  : all            cont:       421
 fixed   :        58                               bin :       421

-------------------------------------------------------------------------------

Objective, min cx
   range: min |c|: 0.00000   min |c|>0: 11.0000     max |c|: 500.000
 distrib:        |c|        vars
                   0         421
           [11, 100)         150
          [100, 500]         271

-------------------------------------------------------------------------------
```

```
Constraint matrix A has
       479 rows (constraints)
       842 columns (variables)
      2091 (0.518449%) nonzero entries (coefficients)

Row nonzeros, A_i
   range: min A_i: 2 (0.23753%)    max A_i: 34 (4.038%)
 distrib:        A_i        rows       rows%        acc%
                   2         421       87.89       87.89
             [8, 15]          20        4.18       92.07
            [16, 31]          30        6.26       98.33
            [32, 34]           8        1.67      100.00

Column nonzeros, A|j
   range: min A|j: 2 (0.417537%)    max A|j: 3 (0.626305%)
 distrib:        A|j        cols       cols%        acc%
                   2         435       51.66       51.66
                   3         407       48.34      100.00

A nonzeros, A(ij)
   range: min |A(ij)|: 1.00000     max |A(ij)|: 100.000
 distrib:      A(ij)      coeffs
             [1, 10)        1670
           [10, 100]         421

-------------------------------------------------------------------------------

Constraint bounds, lb <= Ax <= ub
 distrib:        |b|            lbs             ubs
                   0                            421
             [1, 10]             58              58

Variable bounds, lb <= x <= ub
 distrib:        |b|            lbs             ubs
                   0            842
             [1, 10)                            421
           [10, 100]                            421

-------------------------------------------------------------------------------
```

The survey is divided into six different sections, each described below. To keep the presentation short with focus on key elements the analyzer generally attempts to display information on issues relevant for the current model only: E.g., if the model does not have any conic constraints (this is the case in the example above) or any integer variables, those parts of the analysis will not appear.

12.1.1 General characteristics

The first part of the survey consists of a brief summary of the model's linear and quadratic constraints (indexed by i) and variables (indexed by j). The summary is divided into three subsections:

`Constraints`

`upper bd:` The number of upper bounded constraints, $\sum_{j=0}^{n-1} a_{ij}x_j \leq u_i^c$

`lower bd`: The number of lower bounded constraints, $l_i^c \leq \sum_{j=0}^{n-1} a_{ij}x_j$

`ranged` : The number of ranged constraints, $l_i^c \leq \sum_{j=0}^{n-1} a_{ij}x_j \leq u_i^c$

`fixed` : The number of fixed constraints, $l_i^c = \sum_{j=0}^{n-1} a_{ij}x_j = u_i^c$

`free` : The number of free constraints

`Bounds`

`upper bd`: The number of upper bounded variables, $x_j \leq u_j^x$

`lower bd`: The number of lower bounded variables, $l_k^x \leq x_j$

`ranged` : The number of ranged variables, $l_k^x \leq x_j \leq u_j^x$

`fixed` : The number of fixed variables, $l_k^x = x_j = u_j^x$

`free` : The number of free variables

`Variables`

`cont`: The number of continuous variables, $x_j \in \mathbb{R}$

`bin` : The number of binary variables, $x_j \in \{0, 1\}$

`int` : The number of general integer variables, $x_j \in \mathbb{Z}$

Only constraints, bounds and domains actually in the model will be reported on, cf. appendix J; if all entities in a section turn out to be of the same kind, the number will be replaced by `all` for brevity.

12.1.2 Objective

The second part of the survey focuses on (the linear part of) the objective, summarizing the optimization sense and the coefficients' absolute value range and distribution. The number of 0 (zero) coefficients is singled out (if any such variables are in the problem).

The range is displayed using three terms:

`min |c|`: The minimum absolute value among all coeffecients

`min |c|>0`: The minimum absolute value among the nonzero coefficients

`max |c|`: The maximum absolute value among the coefficients

If some of these extrema turn out to be equal, the display is shortened accordingly:

- If `min |c|` is greater than zero, the `min |c|>0` term is obsolete and will not be displayed
- If only one or two different coefficients occur this will be displayed using `all` and an explicit listing of the coefficients

The absolute value distribution is displayed as a table summarizing the numbers by orders of magnitude (with a ratio of 10). Again, the number of variables with a coefficient of 0 (if any) is singled out. Each line of the table is headed by an interval (half-open intervals including their lower bounds), and is followed by the number of variables with their objective coefficient in this interval. Intervals with no elements are skipped.

12.1.3 Linear constraints

The third part of the survey displays information on the nonzero coefficients of the linear constraint matrix.

Following a brief summary of the matrix dimensions and the number of nonzero coefficients in total, three sections provide further details on how the nonzero coefficients are distributed by row-wise count (`A_i`), by column-wise count (`A|j`), and by absolute value (`|A(ij)|`). Each section is headed by a brief display of the distribution's range (`min` and `max`), and for the row/column-wise counts the corresponding densities are displayed too (in parentheses).

The distribution tables single out three particularly interesting counts: zero, one, and two nonzeros per row/column; the remaining row/column nonzeros are displayed by orders of magnitude (ratio 2). For each interval the relative and accumulated relative counts are also displayed.

Note that constraints may have both linear and quadratic terms, but the empty rows and columns reported in this part of the survey relate to the linear terms only. If empty rows and/or columns are found in the linear constraint matrix, the problem is analyzed further in order to determine if the corresponding constraints have any quadratic terms or the corresponding variables are used in conic or quadratic constraints; cf. the last two examples of appendix J.

The distribution of the absolute values, `|A(ij)|`, is displayed just as for the objective coefficients described above.

12.1.4 Constraint and variable bounds

The fourth part of the survey displays distributions for the absolute values of the finite lower and upper bounds for both constraints and variables. The number of bounds at 0 is singled out and, otherwise, displayed by orders of magnitude (with a ratio of 10).

12.1.5 Quadratic constraints

The fifth part of the survey displays distributions for the nonzero elements in the gradient of the quadratic constraints, i.e. the nonzero row counts for the column vectors Qx. The table is similar to the tables for the linear constraints' nonzero row and column counts described in the survey's third part.

Note: Quadratic constraints may also have a linear part, but that will be included in the linear constraints survey; this means that if a problem has one or more pure quadratic constraints, part three of the survey will report an equal number of linear constraint rows with 0 (zero) nonzeros, cf. the last example in appendix J. Likewise, variables that appear in quadratic terms only will be reported as empty columns (0 nonzeros) in the linear constraint report.

12.1.6 Conic constraints

The last part of the survey summarizes the model's conic constraints. For each of the two types of cones, quadratic and rotated quadratic, the total number of cones are reported, and the distribution of the cones' dimensions are displayed using intervals. Cone dimensions of 2, 3, and 4 are singled out.

12.2 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.

12.2.1 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 formulated as the LP:

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

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:

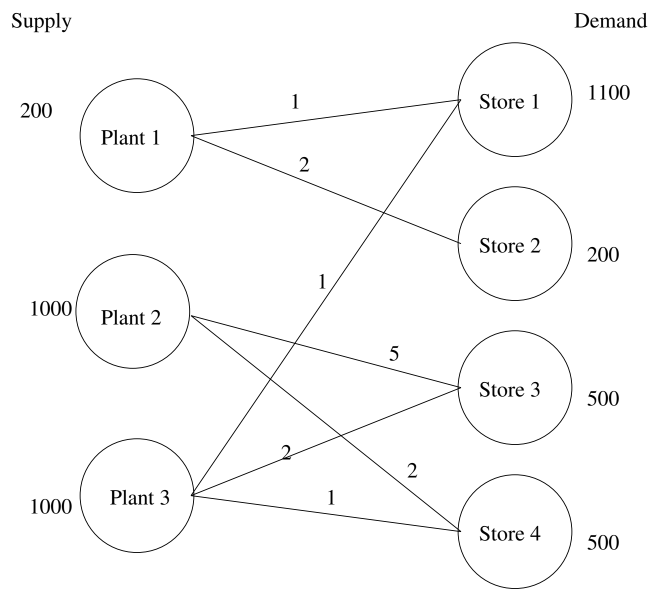

Figure 12.1: Supply, demand and cost of transportation.

```
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.

12.2.2 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 12.2.4) 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.

12.2.3 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.

12.2.3.1 A cautious note

The problem

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

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.

12.2.4 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.

12.2.4.1 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 command line

```
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. This 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**. Since the reduced problem is usually smaller than original problem, it should be easier to locate the cause of the infeasibility in this rather than in the original (12.3).

12.2.4.2 Example: Dual infeasibility

The example problem

```
maximize -  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):

```
The 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 may 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.

12.2.5 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

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

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

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

12.2.6 The certificate 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 certificate 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

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

The well-known Farkas Lemma tells us that (12.8) is infeasible if and only if a certificate 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.

12.2.7 The certificate of dual infeasibility

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

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

with negative objective value, where we use the definitions

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

and

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

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

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

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

Note 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 12.2.2 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:

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

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

Now, the problem

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

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

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

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 may 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 [12] discusses the classical sensitivity analysis in Chapter 10 whereas the book [19, Chapter 19] presents a modern introduction to sensitivity analysis. Finally, it is recommended to read the short paper [21] 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 that we are given the problem

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

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

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

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 that 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.

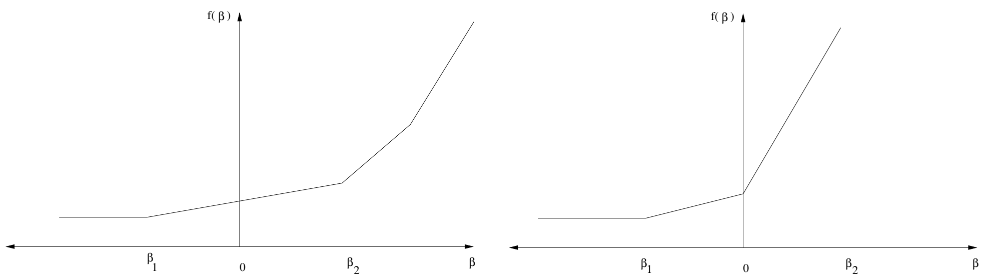

Figure 14.1: The optimal value function $f_{l_i^c}(\beta)$. Left: $\beta = 0$ is in the interior of linearity interval. Right: $\beta = 0$ is a breakpoint.

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. [12, Chapter 10], is based on an optimal basic solution or equivalently on an optimal basis. This method may produce misleading results [19, 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 that 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

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

and

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

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

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

and

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

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

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

and

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

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

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

and

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

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

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

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 that if the upper bound on constraint 1 increases by

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

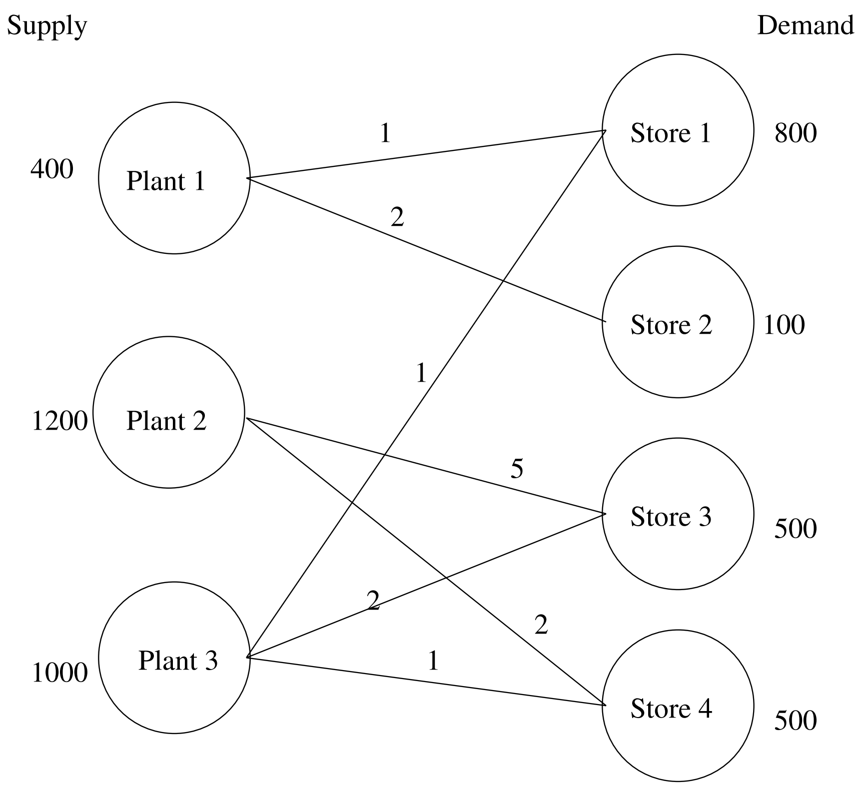

Figure 14.2: Supply, demand and cost of transportation.

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|LU [cname1]
 U|L|LU [cname2]-[cname3]
BOUNDS VARIABLES
 U|L|LU [vname1]
 U|L|LU [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 [18] 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

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

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

  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	Required	Description
`[name]`	5	8	Yes	Name of the `RHS` vector
`[cname1]`	15	8	Yes	Constraint name
`[value1]`	25	12	Yes	Numerical value
`[cname2]`	40	8	No	Constraint name
`[value2]`	50	12	No	Numerical value

The interpretation of a record is that `[name]` is the name of the `RHS` vector to be specified. In general, several vectors can be specified. `[cname1]` denotes a constraint name previously specified in the `ROWS` section. Now, assume that this name has been assigned to the 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 QSECTIONs please note that:

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

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

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

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

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

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

- $\mathbb{R}$ set:

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

- Quadratic cone:

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

- Rotated quadratic cone:

$$\mathcal{C}_t = \left\{ x \in \mathbb{R}^{n^t} : 2x_1x_2 \geq \sum_{j=3}^{n^t} x_j^2,\ x_1, x_2 \geq 0 \right\}. \tag{B.4}$$

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

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

$$x_4 \geq \sqrt{x_5^2 + x_0^2} \tag{B.5}$$

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
*23456789012345678901234567890123456789012345678901234567890
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

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

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

  and it is assumed that

  $$Q^o = (Q^o)^T. \tag{C.1}$$

- $A \in \mathbb{R}^{m \times n}$ is the constraint matrix.
- $l^c \in \mathbb{R}^m$ is the lower limit on the activity for the constraints.

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

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

  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 + 3 y + x^2 + 3 y^2 + z + 1
[/objective]
```

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

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

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

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

```
[tag value] tag with one unnamed argument [/tag]
[tag arg=value] tag with one named argument 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 setting is `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`.

An example of a solution section

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

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

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

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

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:

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

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 converted 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:

$$\begin{array}{lccl} \text{minimize} & 1x_1 + 2x_2 & & \\ \text{subject to} & 2x_3 + 4x_4 & = & 5, \\ & x_5^2 & \leq & 2x_1x_3, \\ & x_6^2 & \leq & 2x_2x_4, \\ & x_5 & = & 1, \\ & x_6 & = & 1, \\ & x \geq 0. & & \end{array} \tag{D.4}$$

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 converted 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:

$$\begin{array}{lccl} \text{maximize} & x_0 + 0.64x_1 & & \\ \text{subject to} & 50x_0 + 31x_1 & \leq & 250, \\ & 3x_0 - 2x_1 & \geq & -4, \\ & x_0, x_1 \geq 0 & & \text{and integer} \end{array} \tag{D.5}$$

This can be implemented in `OPF` with:

```
[comment]
   Written by MOSEK version 5.0.0.7
   Date 20-11-06
   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.

H.1.2 Basis identification parameters.

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_QO_SEPARABLE_REFORMULATION 249
  Determine if Quadratic programing problems should be reformulated to separable form.

H.1.4 Simplex optimizer parameters.

Parameters defining the behavior of the simplex optimizer for linear problems.

- MSK_DPAR_BASIS_REL_TOL_S 182
  Maximum relative dual bound violation allowed in an optimal basic solution.

- MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE 214
  Controls the sign of the columns in the basis matrix corresponding to slack variables.

- MSK_DPAR_BASIS_TOL_S 183
  Maximum absolute dual bound violation in an optimal basic solution.

- MSK_DPAR_BASIS_TOL_X 183
  Maximum absolute primal bound violation allowed in an optimal basic solution.

- MSK_IPAR_LOG_SIM 233
  Controls the amount of log information from the simplex optimizers.

- MSK_IPAR_LOG_SIM_FREQ 234
  Controls simplex logging frequency.

- MSK_IPAR_LOG_SIM_MINOR 234
  Currently not in use.

- MSK_IPAR_SENSITIVITY_OPTIMIZER 256
  Controls which optimizer is used for optimal partition sensitivity analysis.

- MSK_IPAR_SIM_BASIS_FACTOR_USE 257
  Controls whether a (LU) factorization of the basis is used in a hot-start. Forcing a refactorization sometimes improves the stability of the simplex optimizers, but in most cases there is a performance penanlty.

- MSK_IPAR_SIM_DEGEN 257
  Controls how aggressively degeneration is handled.

- MSK_IPAR_SIM_DUAL_PHASEONE_METHOD 258
  An exprimental feature.

- MSK_IPAR_SIM_EXPLOIT_DUPVEC 259
  Controls if the simplex optimizers are allowed to exploit duplicated columns.

- MSK_IPAR_SIM_HOTSTART 259
  Controls the type of hot-start that the simplex optimizer perform.

- MSK_IPAR_SIM_INTEGER 260
  An exprimental feature.

- MSK_DPAR_SIM_LU_TOL_REL_PIV ... 199
  Relative pivot tolerance employed when computing the LU factorization of the basis matrix.
- MSK_IPAR_SIM_MAX_ITERATIONS ... 260
  Maximum number of iterations that can be used by a simplex optimizer.
- MSK_IPAR_SIM_MAX_NUM_SETBACKS ... 261
  Controls how many set-backs that are allowed within a simplex optimizer.
- MSK_IPAR_SIM_NETWORK_DETECT_METHOD ... 262
  Controls which type of detection method the network extraction should use.
- MSK_IPAR_SIM_NON_SINGULAR ... 262
  Controls if the simplex optimizer ensures a non-singular basis, if possible.
- MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD ... 262
  An exprimental feature.
- MSK_IPAR_SIM_REFORMULATION ... 264
  Controls if the simplex optimizers are allowed to reformulate the problem.
- MSK_IPAR_SIM_SAVE_LU ... 264
  Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.
- MSK_IPAR_SIM_SCALING ... 265
  Controls how much effort is used in scaling the problem before a simplex optimizer is used.
- MSK_IPAR_SIM_SCALING_METHOD ... 265
  Controls how the problem is scaled before a simplex optimizer is used.
- MSK_IPAR_SIM_SOLVE_FORM ... 265
  Controls whether the primal or the dual problem is solved by the primal-/dual- simplex optimizer.
- MSK_IPAR_SIM_STABILITY_PRIORITY ... 265
  Controls how high priority the numerical stability should be given.
- MSK_IPAR_SIM_SWITCH_OPTIMIZER ... 266
  Controls the simplex behavior.
- MSK_DPAR_SIMPLEX_ABS_TOL_PIV ... 200
  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 ... 262
  Controls the simplex crash.
- MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION ... 263
  Controls how aggressively restricted selection is used.

H.1.6 Dual simplex optimizer parameters.

Parameters defining the behavior of the dual simplex optimizer for linear problems.

H.1.7 Network simplex optimizer parameters.

Parameters defining the behavior of the network simplex optimizer for linear problems.

H.1.8 Nonlinear convex method parameters.

Parameters defining the behavior of the interior-point method for nonlinear convex problems.

- MSK_DPAR_INTPNT_NL_TOL_PFEAS 189
  Primal feasibility tolerance used when a nonlinear model is solved.

- MSK_DPAR_INTPNT_NL_TOL_REL_GAP 189
  Relative gap termination tolerance for nonlinear problems.

- MSK_DPAR_INTPNT_NL_TOL_REL_STEP 190
  Relative step size to the boundary for general nonlinear optimization problems.

- MSK_DPAR_INTPNT_TOL_INFEAS 190
  Nonlinear solver infeasibility tolerance parameter.

- MSK_IPAR_LOG_CHECK_CONVEXITY 228
  Controls logging in convexity check on quadratic problems. Set to a positive value to turn logging on.

  If a quadratic coefficient matrix is found to violate the requirement of PSD (NSD) then a list of negative (positive) pivot elements is printed. The absolute value of the pivot elements is also shown.

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 186
  Dual feasibility tolerance used by the conic interior-point optimizer.

- MSK_DPAR_INTPNT_CO_TOL_INFEAS 187
  Infeasibility tolerance for the conic solver.

- MSK_DPAR_INTPNT_CO_TOL_MU_RED 187
  Optimality tolerance for the conic solver.

- MSK_DPAR_INTPNT_CO_TOL_NEAR_REL 187
  Optimality tolerance for the conic solver.

- MSK_DPAR_INTPNT_CO_TOL_PFEAS 188
  Primal feasibility tolerance used by the conic interior-point optimizer.

- MSK_DPAR_INTPNT_CO_TOL_REL_GAP 188
  Relative gap termination tolerance used by the conic interior-point optimizer.

H.1.10 The mixed-integer optimization parameters.

- MSK_IPAR_LOG_MIO 231
  Controls the amount of log information from the mixed-integer optimizers.

- MSK_IPAR_LOG_MIO_FREQ 231
  The mixed-integer solver logging frequency.

- MSK_IPAR_MIO_BRANCH_DIR 235
  Controls whether the mixed-integer optimizer is branching up or down by default.

H.1.11 Presolve parameters.

- MSK_IPAR_PRESOLVE_ELIM_FILL 247
  Maximum amount of fill-in in the elimination phase.
- MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES 248
  Control the maximum number of times the eliminator is tried.
- MSK_IPAR_PRESOLVE_ELIMINATOR_USE 248
  Controls whether free or implied free variables are eliminated from the problem.
- MSK_IPAR_PRESOLVE_LEVEL 248
  Currently not used.
- MSK_IPAR_PRESOLVE_LINDEP_USE 248
  Controls whether the linear constraints are checked for linear dependencies.
- MSK_IPAR_PRESOLVE_LINDEP_WORK_LIM 249
  Controls linear dependency check in presolve.
- MSK_DPAR_PRESOLVE_TOL_AIJ 198
  Absolute zero tolerance employed for constraint coefficients in the presolve.
- MSK_DPAR_PRESOLVE_TOL_LIN_DEP 198
  Controls when a constraint is determined to be linearly dependent.
- MSK_DPAR_PRESOLVE_TOL_S 198
  Absolute zero tolerance employed for slack variables in the presolve.
- MSK_DPAR_PRESOLVE_TOL_X 199
  Absolute zero tolerance employed for variables in the presolve.
- MSK_IPAR_PRESOLVE_USE 249
  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 182
  Maximum relative dual bound violation allowed in an optimal basic solution.
- MSK_DPAR_BASIS_TOL_S 183
  Maximum absolute dual bound violation in an optimal basic solution.
- MSK_DPAR_BASIS_TOL_X 183
  Maximum absolute primal bound violation allowed in an optimal basic solution.
- MSK_IPAR_BI_MAX_ITERATIONS 216
  Maximum number of iterations after basis identification.

- MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH 193
  Objective bound.

- MSK_DPAR_MIO_DISABLE_TERM_TIME 193
  Certain termination criteria is disabled within the mixed-integer optimizer for period time specified by the parameter.

- MSK_IPAR_MIO_MAX_NUM_BRANCHES 239
  Maximum number of branches allowed during the branch and bound search.

- MSK_IPAR_MIO_MAX_NUM_SOLUTIONS 240
  Controls how many feasible solutions the mixed-integer optimizer investigates.

- MSK_DPAR_MIO_MAX_TIME 194
  Time limit for the mixed-integer optimizer.

- MSK_DPAR_MIO_NEAR_TOL_REL_GAP 195
  The mixed-integer optimizer is terminated when this tolerance is satisfied.

- MSK_DPAR_MIO_REL_GAP_CONST 195
  This value is used to compute the relative gap for the solution to an integer optimization problem.

- MSK_DPAR_MIO_TOL_REL_GAP 197
  Relative optimality tolerance employed by the mixed-integer optimizer.

- MSK_DPAR_OPTIMIZER_MAX_TIME 198
  Solver time limit.

- MSK_IPAR_SIM_MAX_ITERATIONS 260
  Maximum number of iterations that can be used by a simplex optimizer.

- MSK_DPAR_UPPER_OBJ_CUT 200
  Objective bound.

- MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH 200
  Objective bound.

H.1.13 Progress call-back parameters.

- MSK_DPAR_CALLBACK_FREQ 183
  Controls progress call-back frequency.

- MSK_IPAR_SOLUTION_CALLBACK 267
  Indicates whether solution call-backs will be performed during the optimization.

H.1.14 Non-convex solver parameters.

- MSK_IPAR_LOG_NONCONVEX 231
  Controls amount of output printed by the nonconvex optimizer.

- MSK_IPAR_NONCONVEX_MAX_ITERATIONS 243
  Maximum number of iterations that can be used by the nonconvex optimizer.

- MSK_DPAR_NONCONVEX_TOL_FEAS 197
  Feasibility tolerance used by the nonconvex optimizer.

- MSK_DPAR_NONCONVEX_TOL_OPT 197
  Optimality tolerance used by the nonconvex optimizer.

H.1.15 Feasibility repair parameters.

- MSK_DPAR_FEASREPAIR_TOL 186
  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_AUTO_UPDATE_SOL_INFO 214
  Controls whether the solution information items are automatically updated after an optimization is performed.

- MSK_IPAR_CACHE_LICENSE 216
  Control license caching.

- MSK_IPAR_CACHE_SIZE_L1 216
  Specifies the size of the level 1 cache of the processor.

- MSK_IPAR_CACHE_SIZE_L2 217
  Specifies the size of the level 2 cache of the processor.

- MSK_IPAR_CPU_TYPE 219
  Specifies the CPU type.

- MSK_IPAR_INTPNT_NUM_THREADS 223
  Controls the number of threads employed by the interior-point optimizer. If set to a positive number MOSEK will use this number of threads. If zero the number of threads used will equal the number of cores detected on the machine.

- MSK_IPAR_LICENSE_CACHE_TIME 226
  Setting this parameter no longer has any effect.

- MSK_IPAR_LICENSE_CHECK_TIME 226
  Controls the license manager client behavior.

- MSK_IPAR_LICENSE_WAIT 227
  Controls if MOSEK should queue for a license if none is available.

- MSK_IPAR_LOG_STORAGE 235
  Controls the memory related log information.

H.1.17 Output information parameters.

H.1.18 Extra information about the optimization problem.

H.1.19 Overall solver parameters.

- MSK_IPAR_SENSITIVITY_OPTIMIZER 256
  Controls which optimizer is used for optimal partition sensitivity analysis.
- MSK_IPAR_SENSITIVITY_TYPE 257
  Controls which type of sensitivity analysis is to be performed.
- MSK_IPAR_SOLUTION_CALLBACK 267
  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 213
  Controls how the quadratic matrixes are extended.
- MSK_SPAR_FEASREPAIR_NAME_PREFIX 278
  Feasibility repair name prefix.
- MSK_SPAR_FEASREPAIR_NAME_SEPARATOR 278
  Feasibility repair name separator.
- MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL 278
  Feasibility repair name violation name.
- MSK_IPAR_READ_ADD_ANZ 249
  Controls how the constraint matrix is extended.
- MSK_IPAR_READ_ADD_CON 250
  Additional number of constraints that is made room for in the problem.
- MSK_IPAR_READ_ADD_CONE 250
  Additional number of conic constraints that is made room for in the problem.
- MSK_IPAR_READ_ADD_QNZ 250
  Controls how the quadratic matrixes are extended.
- MSK_IPAR_READ_ADD_VAR 250
  Additional number of variables that is made room for in the problem.
- MSK_IPAR_READ_ANZ 251
  Controls the expected number of constraint non-zeros.
- MSK_IPAR_READ_CON 251
  Controls the expected number of constraints.
- MSK_IPAR_READ_CONE 251
  Controls the expected number of conic constraints.
- MSK_IPAR_READ_QNZ 255
  Controls the expected number of quadratic non-zeros.

- MSK_IPAR_READ_TASK_IGNORE_PARAM 255
  Controls what information is used from the task files.

- MSK_IPAR_READ_VAR 256
  Controls the expected number of variables.

- MSK_IPAR_WRITE_TASK_INC_SOL 275
  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 277
  Name of the bas solution file.

- MSK_SPAR_DATA_FILE_NAME 277
  Data are read and written to this file.

- MSK_SPAR_DEBUG_FILE_NAME 278
  MOSEK debug file.

- MSK_IPAR_INFEAS_REPORT_AUTO 220
  Turns the feasibility report on or off.

- MSK_SPAR_INT_SOL_FILE_NAME 279
  Name of the int solution file.

- MSK_SPAR_ITR_SOL_FILE_NAME 279
  Name of the itr solution file.

- MSK_IPAR_LOG_FILE 230
  If turned on, then some log info is printed when a file is written or read.

- MSK_IPAR_LP_WRITE_IGNORE_INCOMPATIBLE_ITEMS 235
  Controls the result of writing a problem containing incompatible items to an LP file.

- MSK_IPAR_OPF_MAX_TERMS_PER_LINE 244
  The maximum number of terms (linear and quadratic) per line when an OPF file is written.

- MSK_IPAR_OPF_WRITE_HEADER 244
  Write a text header with date and MOSEK version in an OPF file.

- MSK_IPAR_OPF_WRITE_HINTS 244
  Write a hint section with problem dimensions in the beginning of an OPF file.

- MSK_IPAR_OPF_WRITE_PARAMETERS 244
  Write a parameter section in an OPF file.

- MSK_IPAR_OPF_WRITE_PROBLEM 245
  Write objective, constraints, bounds etc. to an OPF file.

- MSK_IPAR_WRITE_SOL_HEAD 274
  Controls solution file format.

- MSK_IPAR_WRITE_SOL_VARIABLES 275
  Controls the solution file format.

- MSK_IPAR_WRITE_TASK_INC_SOL 275
  Controls whether the solutions are stored in the task file too.

- MSK_IPAR_WRITE_XML_MODE 275
  Controls if linear coefficients should be written by row or column when writing in the XML file format.

H.1.22 Analysis parameters.

Parameters controling the behaviour of the problem and solution analyzers.

- MSK_IPAR_ANA_SOL_BASIS 213
  Controls whether the basis matrix is analyzed in solaution analyzer.

- MSK_DPAR_ANA_SOL_INFEAS_TOL 182
  If a constraint violates its bound with an amount larger than this value, the constraint name, index and violation will be printed by the solution analyzer.

- MSK_IPAR_ANA_SOL_PRINT_VIOLATED 213
  Controls whether a list of violated constraints is printed.

H.1.23 Solution input/output parameters.

Parameters defining the behavior of solution reader and writer.

- MSK_SPAR_BAS_SOL_FILE_NAME 277
  Name of the bas solution file.

- MSK_IPAR_INFEAS_REPORT_AUTO 220
  Turns the feasibility report on or off.

- MSK_SPAR_INT_SOL_FILE_NAME 279
  Name of the int solution file.

- MSK_SPAR_ITR_SOL_FILE_NAME 279
  Name of the itr solution file.

- MSK_IPAR_SOL_FILTER_KEEP_BASIC 266
  Controls the license manager client behavior.

- MSK_IPAR_SOL_FILTER_KEEP_RANGED 266
  Control the contents of the solution files.

- MSK_SPAR_SOL_FILTER_XC_LOW 281
  Solution file filter.

H.1.24 Infeasibility report parameters.

- MSK_IPAR_LOG_INFEAS_ANA 230
  Controls log level for the infeasibility analyzer.

H.1.25 License manager parameters.

- MSK_IPAR_LICENSE_ALLOW_OVERUSE 226
  Controls if license overuse is allowed when caching licenses
- MSK_IPAR_LICENSE_CACHE_TIME 226
  Setting this parameter no longer has any effect.
- MSK_IPAR_LICENSE_CHECK_TIME 226
  Controls the license manager client behavior.
- MSK_IPAR_LICENSE_DEBUG 226
  Controls the license manager client debugging behavior.
- MSK_IPAR_LICENSE_PAUSE_TIME 227
  Controls license manager client behavior.
- MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS 227
  Controls license manager client behavior.
- MSK_IPAR_LICENSE_WAIT 227
  Controls if MOSEK should queue for a license if none is available.

H.1.26 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 217
  Specify the level of convexity check on quadratic problems
- MSK_IPAR_CHECK_TASK_DATA 217
  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 184
  Data tolerance threshold.
- MSK_DPAR_DATA_TOL_AIJ_HUGE 184
  Data tolerance threshold.
- MSK_DPAR_DATA_TOL_AIJ_LARGE 184
  Data tolerance threshold.
- MSK_DPAR_DATA_TOL_BOUND_INF 185
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_BOUND_WRN 185
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_C_HUGE 185
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_CJ_LARGE 185
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_QIJ 186
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_X 186
  Data tolerance threshold.

- MSK_IPAR_LOG_CHECK_CONVEXITY 228
  Controls logging in convexity check on quadratic problems. Set to a positive value to turn logging on.

  If a quadratic coefficient matrix is found to violate the requirement of PSD (NSD) then a list of negative (positive) pivot elements is printed. The absolute value of the pivot elements is also shown.

H.1.27 Debugging parameters.

These parameters defines that can be used when debugging a problem.

- MSK_IPAR_AUTO_SORT_A_BEFORE_OPT 214
  Controls whether the elements in each column of A are sorted before an optimization is performed.

- MSK_IPAR_CHECK_TASK_DATA 217
  If this feature is turned on, then the task data is checked for bad values i.e. NaNs. before an optimization is performed.

H.2 Double parameters

- MSK_DPAR_ANA_SOL_INFEAS_TOL 182
  If a constraint violates its bound with an amount larger than this value, the constraint name, index and violation will be printed by the solution analyzer.

- MSK_DPAR_BASIS_REL_TOL_S 182
  Maximum relative dual bound violation allowed in an optimal basic solution.

- MSK_DPAR_BASIS_TOL_S 183
  Maximum absolute dual bound violation in an optimal basic solution.

- MSK_DPAR_BASIS_TOL_X 183
  Maximum absolute primal bound violation allowed in an optimal basic solution.

- MSK_DPAR_CALLBACK_FREQ 183
  Controls progress call-back frequency.

- `ana_sol_infeas_tol`

  Corresponding constant:

  `MSK_DPAR_ANA_SOL_INFEAS_TOL`

  Description:

  If a constraint violates its bound with an amount larger than this value, the constraint name, index and violation will be printed by the solution analyzer.

  Possible Values:

  Any number between 0.0 and +inf.

  Default value:

  `+1e-8`

- `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`

- `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`

- `check_convexity_rel_tol`

  Corresponding constant:
  `MSK_DPAR_CHECK_CONVEXITY_REL_TOL`

Description:

This parameter controls when the full convexity check declares a problem to be non-convex. Increasing this tolerance relaxes the criteria for declaring the problem non-convex.

A problem is declared non-convex if negative (positive) pivot elements are detected in the cholesky factor of a matrix which is required to be PSD (NSD). This parameter controles how much this non-negativity requirement may be violated.

If d_i is the pivot element for column i, then the matrix Q is considered to not be PSD if:

$$d_i \leq -|Q_{ii}| * \texttt{check\_convexity\_rel\_tol}$$

Possible Values:

Any number between 0 and +inf.

Default value:

`1e-10`

- `data_tol_aij`

  Corresponding constant:

  `MSK_DPAR_DATA_TOL_AIJ`

  Description:

  Absolute zero tolerance for elements in A. If any value A_{ij} is smaller than this parameter in absolute terms MOSEK will treat the values as zero and generate a warning.

  Possible Values:

  Any number between 1.0e-16 and 1.0e-6.

  Default value:

  `1.0e-12`

- `data_tol_aij_huge`

  Corresponding constant:

  `MSK_DPAR_DATA_TOL_AIJ_HUGE`

  Description:

  An element in A which is larger than this value in absolute size causes an error.

  Possible Values:

  Any number between 0.0 and +inf.

  Default value:

  `1.0e20`

- `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`

See also:

MSK_DPAR_INTPNT_CO_TOL_NEAR_REL Optimality tolerance for the conic solver.

- `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`

  See also:

  `MSK_DPAR_INTPNT_CO_TOL_NEAR_REL` Optimality tolerance for the conic solver.

- `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`

  See also:

  `MSK_DPAR_INTPNT_CO_TOL_NEAR_REL` Optimality tolerance for the conic solver.

- `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 may be 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 may 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 that 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 either a primal or dual feasible solution is found proving that the optimal objective value is 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

  See also:

  MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH Objective bound.

- 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 criteria are never checked.

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1.0

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 criteria 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-3`

See also:

`MSK_DPAR_MIO_DISABLE_TERM_TIME` Certain termination criteria 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_rel_gap_const`

Corresponding constant:
`MSK_DPAR_MIO_REL_GAP_CONST`

Description:
This value is used to compute the relative gap for the solution to an integer optimization problem.

Possible Values:
Any number between 1.0e-15 and +inf.

Default value:
`1.0e-10`

- `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_feas`

  Corresponding constant:
  `MSK_DPAR_MIO_TOL_FEAS`

  Description:
  Feasibility tolerance for mixed integer solver. Any solution with maximum infeasibility below this value will be considered feasible.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  `1.0e-7`

- `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-4`

- `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 1.0e-15 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

- qcqo_reformulate_rel_drop_tol

Corresponding constant:
MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL

Description:
This parameter determines when columns are dropped in incomplete cholesky factorization doing reformulation of quadratic problems.

Possible Values:
Any number between 0 and +inf.

Default value:
1e-15

- sim_lu_tol_rel_piv

Corresponding constant:
MSK_DPAR_SIM_LU_TOL_REL_PIV

Description:
Relative pivot tolerance employed when computing the LU factorization of the basis in the simplex optimizers and in the basis identification procedure.

A value closer to 1.0 generally improves numerical stability but typically also implies an increase in the computational work.

Possible Values:
Any number between 1.0e-6 and 0.999999.

Default value:
0.01

- 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 either a primal or dual feasible solution is found proving that the optimal objective value is outside, [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

  See also:

  MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH Objective bound.

- 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_OBJECTIVE_SENSE 243
  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 244
  The maximum number of terms (linear and quadratic) per line when an OPF file is written.

- MSK_IPAR_OPF_WRITE_HEADER 244
  Write a text header with date and MOSEK version in an OPF file.

- MSK_IPAR_OPF_WRITE_HINTS 244
  Write a hint section with problem dimensions in the beginning of an OPF file.

- MSK_IPAR_OPF_WRITE_PARAMETERS 244
  Write a parameter section in an OPF file.

- MSK_IPAR_OPF_WRITE_PROBLEM 245
  Write objective, constraints, bounds etc. to an OPF file.

- MSK_IPAR_OPF_WRITE_SOL_BAS 245
  Controls what is written to the OPF files.

- MSK_IPAR_OPF_WRITE_SOL_ITG 245
  Controls what is written to the OPF files.

- MSK_IPAR_OPF_WRITE_SOL_ITR 246
  Controls what is written to the OPF files.

- MSK_IPAR_OPF_WRITE_SOLUTIONS 246
  Enable inclusion of solutions in the OPF files.

- MSK_IPAR_OPTIMIZER 246
  Controls which optimizer is used to optimize the task.

- MSK_IPAR_PARAM_READ_CASE_NAME 247
  If turned on, then names in the parameter file are case sensitive.

- MSK_IPAR_PARAM_READ_IGN_ERROR 247
  If turned on, then errors in paramter settings is ignored.

- MSK_IPAR_PRESOLVE_ELIM_FILL 247
  Maximum amount of fill-in in the elimination phase.

- MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES 248
  Control the maximum number of times the eliminator is tried.

- MSK_IPAR_PRESOLVE_ELIMINATOR_USE 248
  Controls whether free or implied free variables are eliminated from the problem.

- MSK_IPAR_PRESOLVE_LEVEL 248
  Currently not used.

- `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`

- `ana_sol_basis`

  Corresponding constant:
  `MSK_IPAR_ANA_SOL_BASIS`

  Description:
  Controls whether the basis matrix is analyzed in solaution analyzer.

  Possible values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  `MSK_ON`

- `ana_sol_print_violated`

  Corresponding constant:
  `MSK_IPAR_ANA_SOL_PRINT_VIOLATED`

  Description:

  Controls whether a list of violated constraints is printed.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:
MSK_OFF

- auto_sort_a_before_opt

Corresponding constant:
MSK_IPAR_AUTO_SORT_A_BEFORE_OPT

Description:
Controls whether the elements in each column of A are sorted before an optimization is performed. This is not required but makes the optimization more deterministic.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:
MSK_OFF

- auto_update_sol_info

Corresponding constant:
MSK_IPAR_AUTO_UPDATE_SOL_INFO

Description:
Controls whether the solution information items are automatically updated after an optimization is performed.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:
MSK_ON

- basis_solve_use_plus_one

Corresponding constant:
MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE

Description:
If a slack variable is in the basis, then the corresponding column in the basis is a unit vector with -1 in the right position. However, if this parameter is set to MSK_ON, -1 is replaced by 1.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:
MSK_OFF

- 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_PRIMAL_DUAL_SIMPLEX The primal dual simplex optimizer is used.

MSK_OPTIMIZER_CONIC The optimizer for problems having conic constraints.

MSK_OPTIMIZER_NONCONVEX The optimizer for nonconvex nonlinear problems.

MSK_OPTIMIZER_QCONE For internal use only.

MSK_OPTIMIZER_PRIMAL_SIMPLEX The primal simplex optimizer is used.

MSK_OPTIMIZER_FREE_SIMPLEX One of the simplex optimizers 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_license

Corresponding constant:

MSK_IPAR_CACHE_LICENSE

Description:

Specifies if the license is kept checked out for the lifetime of the mosek environment (on) or returned to the server immediately after the optimization (off).

Check-in and check-out of licenses have an overhead. Frequent communication with the license server should be avoided.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- 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.

MSK_CHECK_CONVEXITY_FULL Perform a full convexity check.

Default value:
MSK_CHECK_CONVEXITY_FULL

- 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_ON

- `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

- 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. If set to a positive number MOSEK will use this number of threads. If zero the number of threads used will equal the number of cores detected on the machine.

  Possible Values:
  Any integer greater or equal to 0.

  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_GUESS The optimizer guesses a starting point.

MSK_STARTING_POINT_SATISFY_BOUNDS The starting point is choosen to satisfy all the simple bounds on nonlinear variables. If this starting point is employed, then more care than usual should employed when choosing the bounds on the nonlinear variables. In particular very tight bounds should be avoided.

MSK_STARTING_POINT_CONSTANT The optimizer constructs a starting point by assigning a constant value to all primal and dual variables. This starting point is normally robust.

MSK_STARTING_POINT_FREE The starting point is chosen automatically.

Default value:
MSK_STARTING_POINT_FREE

- lic_trh_expiry_wrn

Corresponding constant:
MSK_IPAR_LIC_TRH_EXPIRY_WRN

Description:
If a license feature expires in a numbers days less than the value of this parameter then a warning will be issued.

Possible Values:
Any number between 0 and +inf.

Default value:
`7`

- `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:
Setting this parameter no longer has any effect. Please see `MSK_IPAR_CACHE_LICENSE` for an alternative.

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 milliseconds between each check of whether a license has 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_check_convexity

  Corresponding constant:
  MSK_IPAR_LOG_CHECK_CONVEXITY

Description:
Controls logging in convexity check on quadratic problems. Set to a positive value to turn logging on.

If a quadratic coefficient matrix is found to violate the requirement of PSD (NSD) then a list of negative (positive) pivot elements is printed. The absolute value of the pivot elements is also shown.

Possible Values:
Any number between 0 and +inf.

Default value:
`0`

- `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:
`1000`

- `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`

- `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-integer 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 mixed integer solution exists, then MOSEK generates an initial solution to the mixed integer problem by fixing all integer values and solving the remaining problem.

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 mixed integer 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_hotstart`

Corresponding constant:

`MSK_IPAR_MIO_HOTSTART`

Description:

Controls whether the integer optimizer is hot-started.

Possible values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

`MSK_ON`

- `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:

Controls the size of the local search space when doing local branching.

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 criteria 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 criteria 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 criteria 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_PRIMAL_DUAL_SIMPLEX The primal dual simplex optimizer is used.

MSK_OPTIMIZER_CONIC The optimizer for problems having conic constraints.

MSK_OPTIMIZER_NONCONVEX The optimizer for nonconvex nonlinear problems.

MSK_OPTIMIZER_QCONE For internal use only.

MSK_OPTIMIZER_PRIMAL_SIMPLEX The primal simplex optimizer is used.

MSK_OPTIMIZER_FREE_SIMPLEX One of the simplex optimizers 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_optimizer_mode

Corresponding constant:
MSK_IPAR_MIO_OPTIMIZER_MODE

Description:
An exprimental feature.

Possible Values:
Any number between 0 and 1.

Default value:
0

- 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_PRIMAL_DUAL_SIMPLEX The primal dual simplex optimizer is used.

MSK_OPTIMIZER_CONIC The optimizer for problems having conic constraints.

MSK_OPTIMIZER_NONCONVEX The optimizer for nonconvex nonlinear problems.

MSK_OPTIMIZER_QCONE For internal use only.

MSK_OPTIMIZER_PRIMAL_SIMPLEX The primal simplex optimizer is used.

MSK_OPTIMIZER_FREE_SIMPLEX One of the simplex optimizers 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:
The paramter 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_PRIMAL_DUAL_SIMPLEX The primal dual simplex optimizer is used.

MSK_OPTIMIZER_CONIC The optimizer for problems having conic constraints.

MSK_OPTIMIZER_NONCONVEX The optimizer for nonconvex nonlinear problems.

MSK_OPTIMIZER_QCONE For internal use only.

MSK_OPTIMIZER_PRIMAL_SIMPLEX The primal simplex optimizer is used.

MSK_OPTIMIZER_FREE_SIMPLEX One of the simplex optimizers 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_max_num_tries`

  Corresponding constant:
  `MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES`

  Description:
  Control the maximum number of times the eliminator is tried.

  Possible Values:
  A negative value implies MOSEK decides maximum number of times.

  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`

- `qo_separable_reformulation`

  Corresponding constant:
  `MSK_IPAR_QO_SEPARABLE_REFORMULATION`

  Description:
  Determine if Quadratic programing problems should be reformulated to separable form.

  Possible values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  `MSK_OFF`

- `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_FREE_MPS` The data data a free MPS 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 mixed integer constraints are ignored when a problem is 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 characters 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_PRIMAL_DUAL_SIMPLEX The primal dual simplex optimizer is used.

MSK_OPTIMIZER_CONIC The optimizer for problems having conic constraints.

MSK_OPTIMIZER_NONCONVEX The optimizer for nonconvex nonlinear problems.

MSK_OPTIMIZER_QCONE For internal use only.

MSK_OPTIMIZER_PRIMAL_SIMPLEX The primal simplex optimizer is used.

MSK_OPTIMIZER_FREE_SIMPLEX One of the simplex optimizers 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_basis_factor_use

Corresponding constant:
MSK_IPAR_SIM_BASIS_FACTOR_USE

Description:
Controls whether a (LU) factorization of the basis is used in a hot-start. Forcing a refactorization sometimes improves the stability of the simplex optimizers, but in most cases there is a performance penanlty.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:
MSK_ON

- sim_degen

Corresponding constant:
MSK_IPAR_SIM_DEGEN

Description:
Controls how aggressively degeneration is handled.

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_phaseone_method

Corresponding constant:
MSK_IPAR_SIM_DUAL_PHASEONE_METHOD

Description:
An exprimental feature.

Possible Values:
Any number between 0 and 10.

Default value:
0

- 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_exploit_dupvec`

Corresponding constant:
`MSK_IPAR_SIM_EXPLOIT_DUPVEC`

Description:
Controls if the simplex optimizers are allowed to exploit duplicated columns.

Possible values:

`MSK_SIM_EXPLOIT_DUPVEC_ON` Allow the simplex optimizer to exploit duplicated columns.

`MSK_SIM_EXPLOIT_DUPVEC_OFF` Disallow the simplex optimizer to exploit duplicated columns.

`MSK_SIM_EXPLOIT_DUPVEC_FREE` The simplex optimizer can choose freely.

Default value:
`MSK_SIM_EXPLOIT_DUPVEC_OFF`

- `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_hotstart_lu

Corresponding constant:
MSK_IPAR_SIM_HOTSTART_LU

Description:
Determines if the simplex optimizer should exploit the initial factorization.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:
MSK_ON

- sim_integer

Corresponding constant:
MSK_IPAR_SIM_INTEGER

Description:
An exprimental feature.

Possible Values:
Any number between 0 and 10.

Default value:
0

- 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 set-backs are allowed within a simplex optimizer. A set-back 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_phaseone_method

Corresponding constant:
MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD

Description:
An exprimental feature.

Possible Values:
Any number between 0 and 10.

Default value:
0

- 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_reformulation

Corresponding constant:
MSK_IPAR_SIM_REFORMULATION

Description:
Controls if the simplex optimizers are allowed to reformulate the problem.

Possible values:

MSK_SIM_REFORMULATION_ON Allow the simplex optimizer to reformulate the problem.

MSK_SIM_REFORMULATION_AGGRESSIVE The simplex optimizer should use an aggressive reformulation strategy.

MSK_SIM_REFORMULATION_OFF Disallow the simplex optimizer to reformulate the problem.

MSK_SIM_REFORMULATION_FREE The simplex optimizer can choose freely.

Default value:
MSK_SIM_REFORMULATION_OFF

- 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 much effort is used in scaling the problem 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_scaling_method

Corresponding constant:
MSK_IPAR_SIM_SCALING_METHOD

Description:
Controls how the problem is scaled before a simplex optimizer is used.

Possible values:

MSK_SCALING_METHOD_POW2 Scales only with power of 2 leaving the mantissa untouched.

MSK_SCALING_METHOD_FREE The optimizer chooses the scaling heuristic.

Default value:
MSK_SCALING_METHOD_POW2

- 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

- timing_level

Corresponding constant:
MSK_IPAR_TIMING_LEVEL

Description:
Controls the a amount of timing performed inside MOSEK.

Possible Values:
Any integer greater or equal to 0.

Default value:
1

- 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_FREE_MPS The data data a free MPS 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

- `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 Function opcode

Value	Name Description
1	MSK_ADOP_SUB Subtract two operands.
4	MSK_ADOP_POW First operand to the power the second operand.
7	MSK_ADOP_RET Return one operand.
0	MSK_ADOP_ADD Add two operands.
5	MSK_ADOP_EXP Exponential function of one oparand.
2	MSK_ADOP_MUL Multiply two operands.
3	MSK_ADOP_DIV Divide two operands.
6	MSK_ADOP_LOG

continued on next page

continued from previous page

	Logarithm function of one operand.

I.3 Function operand type

Value	Name Description
2	MSK_ADOPTYPE_VARIABLE Operand refers to a variable.
0	MSK_ADOPTYPE_NONE Operand not used.
1	MSK_ADOPTYPE_CONSTANT Operand refers to a constant.
3	MSK_ADOPTYPE_REFERENCE Operand refers to the result of another operation.

I.4 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.5 Bound keys

Value	Name Description
2	MSK_BK_FX

continued on next page

continued from previous page

	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.6 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-integer optimizer decides which branch to choose.

I.7 Progress call-back codes

Value	Name Description
44	MSK_CALLBACK_END_INTPNT The call-back function is called when the interior-point optimizer is terminated.
21	MSK_CALLBACK_BEGIN_PRIMAL_DUAL_SIMPLEX_BI The call-back function is called from within the basis identification procedure when the primal-dual simplex clean-up phase is started.
48	MSK_CALLBACK_END_NETWORK_PRIMAL_SIMPLEX The call-back function is called when the primal network simplex optimizer is terminated.
99	MSK_CALLBACK_READ_ADD_CONS A chunk of constraints has been read from a problem file.
115	MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX_BI

continued on next page

continued from previous page

	The call-back function is called from within the basis identification procedure at an intermediate point in the primal simplex clean-up phase. The frequency of the call-backs is controlled by the MSK_IPAR_LOG_SIM_FREQ parameter.
46	MSK_CALLBACK_END_MIO The call-back function is called when the mixed-integer optimizer is terminated.
13	MSK_CALLBACK_BEGIN_NETWORK_DUAL_SIMPLEX The call-back function is called when the dual network simplex optimizer is started.
35	MSK_CALLBACK_END_CONCURRENT Concurrent optimizer is terminated.
93	MSK_CALLBACK_NEW_INT_MIO The call-back function is called after a new integer solution has been located by the mixed-integer optimizer.
88	MSK_CALLBACK_IM_PRIMAL_SIMPLEX The call-back function is called at an intermediate point in the primal simplex optimizer.
64	MSK_CALLBACK_END_SIMPLEX_NETWORK_DETECT The call-back function is called when the network detection procedure is terminated.
47	MSK_CALLBACK_END_NETWORK_DUAL_SIMPLEX The call-back function is called when the dual network simplex optimizer is terminated.
72	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_DUAL_BI The call-back function is called from within the basis identification procedure at an intermediate point in the dual phase.
8	MSK_CALLBACK_BEGIN_FULL_CONVEXITY_CHECK Begin full convexity check.
3	MSK_CALLBACK_BEGIN_DUAL_BI The call-back function is called from within the basis identification procedure when the dual phase is started.
4	MSK_CALLBACK_BEGIN_DUAL_SENSITIVITY Dual sensitivity analysis is started.
79	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.
19	MSK_CALLBACK_BEGIN_PRIMAL_BI The call-back function is called from within the basis identification procedure when the primal phase is started.

continued on next page

continued from previous page	
100	MSK_CALLBACK_READ_ADD_QNZ A chunk of Q non-zeos has been read from a problem file.
1	MSK_CALLBACK_BEGIN_CONCURRENT Concurrent optimizer is started.
104	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.
70	MSK_CALLBACK_IM_DUAL_SIMPLEX The call-back function is called at an intermediate point in the dual simplex optimizer.
11	MSK_CALLBACK_BEGIN_LICENSE_WAIT Begin waiting for license.
81	MSK_CALLBACK_IM_NETWORK_PRIMAL_SIMPLEX The call-back function is called at an intermediate point in the primal network simplex optimizer.
49	MSK_CALLBACK_END_NETWORK_SIMPLEX The call-back function is called when the simplex network optimizer is terminated.
32	MSK_CALLBACK_CONIC The call-back function is called from within the conic optimizer after the information database has been updated.
89	MSK_CALLBACK_IM_QO_REFORMULATE The call-back function is called at an intermediate stage of the QP to SOCP reformulation.
2	MSK_CALLBACK_BEGIN_CONIC The call-back function is called when the conic optimizer is started.
106	MSK_CALLBACK_UPDATE_DUAL_SIMPLEX_BI The call-back function is called from within the basis identification procedure at an intermediate point in the dual simplex clean-up phase. The frequency of the call-backs is controlled by the MSK_IPAR_LOG_SIM_FREQ parameter.
51	MSK_CALLBACK_END_OPTIMIZER The call-back function is called when the optimizer is terminated.
110	MSK_CALLBACK_UPDATE_PRESOLVE The call-back function is called from within the presolve procedure.
90	MSK_CALLBACK_IM_SIMPLEX The call-back function is called from within the simplex optimizer at an intermediate point.
102	MSK_CALLBACK_READ_OPF The call-back function is called from the OPF reader.
73	MSK_CALLBACK_IM_LICENSE_WAIT MOSEK is waiting for a license.
15	MSK_CALLBACK_BEGIN_NETWORK_SIMPLEX
	continued on next page

continued from previous page

	The call-back function is called when the simplex network optimizer is started.
36	MSK_CALLBACK_END_CONIC
	The call-back function is called when the conic optimizer is terminated.
107	MSK_CALLBACK_UPDATE_NETWORK_DUAL_SIMPLEX
	The call-back function is called in the dual network simplex optimizer.
26	MSK_CALLBACK_BEGIN_QCQO_REFORMULATE
	Begin QCQO reformulation.
38	MSK_CALLBACK_END_DUAL_SENSITIVITY
	Dual sensitivity analysis is terminated.
59	MSK_CALLBACK_END_PRIMAL_SIMPLEX_BI
	The call-back function is called from within the basis identification procedure when the primal clean-up phase is terminated.
101	MSK_CALLBACK_READ_ADD_VARS
	A chunk of variables has been read from a problem file.
103	MSK_CALLBACK_READ_OPF_SECTION
	A chunk of Q non-zeos has been read from a problem file.
74	MSK_CALLBACK_IM_LU
	The call-back function is called from within the LU factorization procedure at an intermediate point.
41	MSK_CALLBACK_END_DUAL_SIMPLEX_BI
	The call-back function is called from within the basis identification procedure when the dual clean-up phase is terminated.
45	MSK_CALLBACK_END_LICENSE_WAIT
	End waiting for license.
84	MSK_CALLBACK_IM_PRESOLVE
	The call-back function is called from within the presolve procedure at an intermediate stage.
5	MSK_CALLBACK_BEGIN_DUAL_SETUP_BI
	The call-back function is called when the dual BI phase is started.
43	MSK_CALLBACK_END_INFEAS_ANA
	The call-back function is called when the infeasibility analyzer is terminated.
92	MSK_CALLBACK_INTPNT
	The call-back function is called from within the interior-point optimizer after the information database has been updated.
111	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.
94	MSK_CALLBACK_NONCOVEX
	The call-back function is called from within the nonconvex optimizer after the information database has been updated.
113	MSK_CALLBACK_UPDATE_PRIMAL_DUAL_SIMPLEX_BI

continued on next page

continued from previous page

	The call-back function is called from within the basis identification procedure at an intermediate point in the primal-dual simplex clean-up phase. The frequency of the call-backs is controlled by the MSK_IPAR_LOG_SIM_FREQ parameter.
109	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.
37	MSK_CALLBACK_END_DUAL_BI The call-back function is called from within the basis identification procedure when the dual phase is terminated.
61	MSK_CALLBACK_END_READ MOSEK has finished reading a problem file.
98	MSK_CALLBACK_READ_ADD_CONES A chunk of cones has been read from a problem file.
25	MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX_BI The call-back function is called from within the basis identification procedure when the primal simplex clean-up phase is started.
30	MSK_CALLBACK_BEGIN_SIMPLEX_NETWORK_DETECT The call-back function is called when the network detection procedure is started.
97	MSK_CALLBACK_READ_ADD_ANZ A chunk of A non-zeos has been read from a problem file.
86	MSK_CALLBACK_IM_PRIMAL_DUAL_SIMPLEX The call-back function is called at an intermediate point in the primal-dual simplex optimizer.
114	MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX The call-back function is called in the primal simplex optimizer.
33	MSK_CALLBACK_DUAL_SIMPLEX The call-back function is called from within the dual simplex optimizer.
71	MSK_CALLBACK_IM_FULL_CONVEXITY_CHECK The call-back function is called at an intermediate stage of the full convexity check.
95	MSK_CALLBACK_PRIMAL_SIMPLEX The call-back function is called from within the primal simplex optimizer.
16	MSK_CALLBACK_BEGIN_NONCONVEX The call-back function is called when the nonconvex optimizer is started.
91	MSK_CALLBACK_IM_SIMPLEX_BI

continued on next page

continued from previous page

	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.
6	MSK_CALLBACK_BEGIN_DUAL_SIMPLEX The call-back function is called when the dual simplex optimizer started.
24	MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX The call-back function is called when the primal simplex optimizer is started.
50	MSK_CALLBACK_END_NONCONVEX The call-back function is called when the nonconvex optimizer is terminated.
23	MSK_CALLBACK_BEGIN_PRIMAL_SETUP_BI The call-back function is called when the primal BI setup is started.
17	MSK_CALLBACK_BEGIN_OPTIMIZER The call-back function is called when the optimizer is started.
27	MSK_CALLBACK_BEGIN_READ MOSEK has started reading a problem file.
82	MSK_CALLBACK_IM_NONCONVEX The call-back function is called at an intermediate stage within the nonconvex optimizer where the information database has not been updated.
58	MSK_CALLBACK_END_PRIMAL_SIMPLEX The call-back function is called when the primal simplex optimizer is terminated.
55	MSK_CALLBACK_END_PRIMAL_DUAL_SIMPLEX_BI The call-back function is called from within the basis identification procedure when the primal-dual clean-up phase is terminated.
66	MSK_CALLBACK_IM_BI The call-back function is called from within the basis identification procedure at an intermediate point.
80	MSK_CALLBACK_IM_NETWORK_DUAL_SIMPLEX The call-back function is called at an intermediate point in the dual network simplex optimizer.
39	MSK_CALLBACK_END_DUAL_SETUP_BI The call-back function is called when the dual BI phase is terminated.
34	MSK_CALLBACK_END_BI The call-back function is called when the basis identification procedure is terminated.
57	MSK_CALLBACK_END_PRIMAL_SETUP_BI The call-back function is called when the primal BI setup is terminated.
31	MSK_CALLBACK_BEGIN_WRITE

continued on next page

continued from previous page

	MOSEK has started writing a problem file.
63	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.
56	MSK_CALLBACK_END_PRIMAL_SENSITIVITY Primal sensitivity analysis is terminated.
28	MSK_CALLBACK_BEGIN_SIMPLEX The call-back function is called when the simplex optimizer is started.
52	MSK_CALLBACK_END_PRESOLVE The call-back function is called when the presolve is completed.
96	MSK_CALLBACK_QCONE The call-back function is called from within the Qcone optimizer.
9	MSK_CALLBACK_BEGIN_INFEAS_ANA The call-back function is called when the infeasibility analyzer is started.
20	MSK_CALLBACK_BEGIN_PRIMAL_DUAL_SIMPLEX The call-back function is called when the primal-dual simplex optimizer is started.
22	MSK_CALLBACK_BEGIN_PRIMAL_SENSITIVITY Primal sensitivity analysis is started.
7	MSK_CALLBACK_BEGIN_DUAL_SIMPLEX_BI The call-back function is called from within the basis identification procedure when the dual simplex clean-up phase is started.
60	MSK_CALLBACK_END_QCQO_REFORMULATE End QCQO reformulation.
87	MSK_CALLBACK_IM_PRIMAL_SENSIVITY The call-back function is called at an intermediate stage of the primal sensitivity analysis.
65	MSK_CALLBACK_END_WRITE MOSEK has finished writing a problem file.
40	MSK_CALLBACK_END_DUAL_SIMPLEX The call-back function is called when the dual simplex optimizer is terminated.
112	MSK_CALLBACK_UPDATE_PRIMAL_DUAL_SIMPLEX The call-back function is called in the primal-dual simplex optimizer.
29	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.
10	MSK_CALLBACK_BEGIN_INTPNT The call-back function is called when the interior-point optimizer is started.
69	MSK_CALLBACK_IM_DUAL_SENSIVITY The call-back function is called at an intermediate stage of the dual sensitivity analysis.

continued on next page

continued from previous page

62	MSK_CALLBACK_END_SIMPLEX The call-back function is called when the simplex optimizer is terminated.
53	MSK_CALLBACK_END_PRIMAL_BI The call-back function is called from within the basis identification procedure when the primal phase is terminated.
75	MSK_CALLBACK_IM_MIO The call-back function is called at an intermediate point in the mixed-integer optimizer.
105	MSK_CALLBACK_UPDATE_DUAL_SIMPLEX The call-back function is called in the dual simplex optimizer.
77	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.
54	MSK_CALLBACK_END_PRIMAL_DUAL_SIMPLEX The call-back function is called when the primal-dual simplex optimizer is terminated.
67	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.
78	MSK_CALLBACK_IM_MIO_PRESOLVE The call-back function is called at an intermediate point in the mixed-integer optimizer while running the presolve.
0	MSK_CALLBACK_BEGIN_BI The basis identification procedure has been started.
76	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.
116	MSK_CALLBACK_WRITE_OPF The call-back function is called from the OPF writer.
108	MSK_CALLBACK_UPDATE_NETWORK_PRIMAL_SIMPLEX The call-back function is called in the primal network simplex optimizer.
42	MSK_CALLBACK_END_FULL_CONVEXITY_CHECK End full convexity check.
83	MSK_CALLBACK_IM_ORDER The call-back function is called from within the matrix ordering procedure at an intermediate point.
85	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.
18	MSK_CALLBACK_BEGIN_PRESOLVE The call-back function is called when the presolve is started.
12	MSK_CALLBACK_BEGIN_MIO

continued on next page

continued from previous page	
	The call-back function is called when the mixed-integer optimizer is started.
14	`MSK_CALLBACK_BEGIN_NETWORK_PRIMAL_SIMPLEX`
	The call-back function is called when the primal network simplex optimizer is started.

I.8 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.
2	`MSK_CHECK_CONVEXITY_FULL`
	Perform a full convexity check.

I.9 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.10 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.11 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.12 Data format types

Value	Name Description
5	MSK_DATA_FORMAT_XML The data file is an XML formatted file.
6	MSK_DATA_FORMAT_FREE_MPS The data data a free MPS 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.13 Double information items

Value	Name Description
13	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).
58	MSK_DINF_SOL_ITR_MAX_PCNI Maximal primal cone infeasibility in the interior-point solution. Updated at the end of the optimization.
32	MSK_DINF_RD_TIME Time spent reading the data file.
28	MSK_DINF_PRESOLVE_ELI_TIME Total time spent in the eliminator since the presolve was invoked.
22	MSK_DINF_MIO_OPTIMIZER_TIME Time spent in the optimizer while solving the relaxtions.
10	MSK_DINF_INTPNT_FACTOR_NUM_FLOPS An estimate of the number of flops used in the factorization.
25	MSK_DINF_MIO_TIME Time spent in the mixed-integer optimizer.
4	MSK_DINF_BI_DUAL_TIME Time spent within the dual phase basis identification procedure since its invocation.
37	MSK_DINF_SIM_NETWORK_TIME Time spent in the network simplex optimizer since invoking it.
30	MSK_DINF_PRESOLVE_TIME Total time (in seconds) spent in the presolve since it was invoked.
29	MSK_DINF_PRESOLVE_LINDEP_TIME Total time spent in the linear dependency checker since the presolve was invoked.
33	MSK_DINF_SIM_DUAL_TIME Time spent in the dual simplex optimizer since invoking it.
60	MSK_DINF_SOL_ITR_MAX_PINTI Maximal primal integer infeasibility in the interior-point solution. Updated at the end of the optimization.
38	MSK_DINF_SIM_OBJ Objective value reported by the simplex optimizer.
21	MSK_DINF_MIO_OBJ_REL_GAP

continued on next page

continued from previous page

	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 $$\frac{\lvert(\text{objective value of feasible solution}) - (\text{objective bound})\rvert}{\max(\delta, \lvert(\text{objective value of feasible solution})\rvert)}.$$ where δ is given by the paramater MSK_DPAR_MIO_REL_GAP_CONST. Otherwise it has the value -1.0.
48	MSK_DINF_SOL_BAS_PRIMAL_OBJ
	Primal objective value of the basic solution. Updated at the end of the optimization.
17	MSK_DINF_MIO_HEURISTIC_TIME
	Time spent in the optimizer while solving the relaxtions.
57	MSK_DINF_SOL_ITR_MAX_PBI
	Maximal primal bound infeasibility in the interior-point solution. Updated at the end of the optimization.
45	MSK_DINF_SOL_BAS_MAX_PBI
	Maximal primal bound infeasibility in the basic solution. Updated at the end of the optimization.
27	MSK_DINF_OPTIMIZER_TIME
	Total time spent in the optimizer since it was invoked.
55	MSK_DINF_SOL_ITR_MAX_DCNI
	Maximal dual cone infeasibility in the interior-point solution. Updated at the end of the optimization.
24	MSK_DINF_MIO_ROOT_PRESOLVE_TIME
	Time spent in while presolveing the root relaxation.
9	MSK_DINF_INTPNT_DUAL_OBJ
	Dual objective value reported by the interior-point or Qcone optimizer.
15	MSK_DINF_INTPNT_TIME
	Time spent within the interior-point optimizer since its invocation.
16	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.
34	MSK_DINF_SIM_FEAS
	Feasibility measure reported by the simplex optimizer.
56	MSK_DINF_SOL_ITR_MAX_DEQI
	Maximal dual equality infeasibility in the interior-point solution. Updated at the end of the optimization.
40	MSK_DINF_SIM_PRIMAL_TIME
	Time spent in the primal simplex optimizer since invoking it.
41	MSK_DINF_SIM_TIME

continued on next page

continued from previous page

	Time spent in the simplex optimizer since invoking it.
36	MSK_DINF_SIM_NETWORK_PRIMAL_TIME
	Time spent in the primal network simplex optimizer since invoking it.
47	MSK_DINF_SOL_BAS_MAX_PINTI
	Maximal primal integer infeasibility in the basic solution. Updated at the end of the optimization.
51	MSK_DINF_SOL_INT_MAX_PINTI
	Maximal primal integer infeasibility in the integer solution. Updated at the end of the optimization.
3	MSK_DINF_BI_CLEAN_TIME
	Time spent within the clean-up phase of the basis identification procedure since its invocation.
31	MSK_DINF_QCQO_REFORMULATE_TIME
	Time spent with QP reformulation.
53	MSK_DINF_SOL_ITR_DUAL_OBJ
	Dual objective value of the interior-point solution. Updated at the end of the optimization.
49	MSK_DINF_SOL_INT_MAX_PBI
	Maximal primal bound infeasibility in the integer solution. Updated at the end of the optimization.
8	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.)
7	MSK_DINF_CONCURRENT_TIME
	Time spent within the concurrent optimizer since its invocation.
11	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.
50	MSK_DINF_SOL_INT_MAX_PEQI
	Maximal primal equality infeasibility in the basic solution. Updated at the end of the optimization.
61	MSK_DINF_SOL_ITR_PRIMAL_OBJ
	Primal objective value of the interior-point solution. Updated at the end of the optimization.
35	MSK_DINF_SIM_NETWORK_DUAL_TIME
	Time spent in the dual network simplex optimizer since invoking it.
20	MSK_DINF_MIO_OBJ_INT

continued on next page

continued from previous page

	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.
26	MSK_DINF_MIO_USER_OBJ_CUT If the objective cut is used, then this information item has the value of the cut.
43	MSK_DINF_SOL_BAS_MAX_DBI Maximal dual bound infeasibility in the basic solution. Updated at the end of the optimization.
46	MSK_DINF_SOL_BAS_MAX_PEQI Maximal primal equality infeasibility in the basic solution. Updated at the end of the optimization.
19	MSK_DINF_MIO_OBJ_BOUND The best known bound on the objective function. This value is undefined until at least one relaxation has been solved: To see if this is the case check that MSK_IINF_MIO_NUM_RELAX is stricly positive.
0	MSK_DINF_BI_CLEAN_DUAL_TIME Time spent within the dual clean-up optimizer of the basis identification procedure since its invocation.
6	MSK_DINF_BI_TIME Time spent within the basis identification procedure since its invocation.
42	MSK_DINF_SOL_BAS_DUAL_OBJ Dual objective value of the basic solution. Updated at the end of the optimization.
1	MSK_DINF_BI_CLEAN_PRIMAL_DUAL_TIME Time spent within the primal-dual clean-up optimizer of the basis identification procedure since its invocation.
14	MSK_DINF_INTPNT_PRIMAL_OBJ Primal objective value reported by the interior-point or Qcone optimizer.
12	MSK_DINF_INTPNT_ORDER_TIME Order time (in seconds).
52	MSK_DINF_SOL_INT_PRIMAL_OBJ Primal objective value of the integer solution. Updated at the end of the optimization.
5	MSK_DINF_BI_PRIMAL_TIME Time spent within the primal phase of the basis identification procedure since its invocation.
18	MSK_DINF_MIO_OBJ_ABS_GAP

continued on next page

continued from previous page

	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.
44	MSK_DINF_SOL_BAS_MAX_DEQI Maximal dual equality infeasibility in the basic solution. Updated at the end of the optimization.
59	MSK_DINF_SOL_ITR_MAX_PEQI Maximal primal equality infeasibility in the interior-point solution. Updated at the end of the optimization.
39	MSK_DINF_SIM_PRIMAL_DUAL_TIME Time spent in the primal-dual simplex optimizer optimizer since invoking it.
2	MSK_DINF_BI_CLEAN_PRIMAL_TIME Time spent within the primal clean-up optimizer of the basis identification procedure since its invocation.
54	MSK_DINF_SOL_ITR_MAX_DBI Maximal dual bound infeasibility in the interior-point solution. Updated at the end of the optimization.
23	MSK_DINF_MIO_ROOT_OPTIMIZER_TIME Time spent in the optimizer while solving the root relaxation.

I.14 Double parameters

Value	Name Description
40	MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH 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 $-\infty$.
43	MSK_DPAR_MIO_MAX_TIME This parameter limits the maximum time spent by the mixed-integer optimizer. A negative number means infinity.
2	MSK_DPAR_BASIS_TOL_S Maximum absolute dual bound violation in an optimal basic solution.
60	MSK_DPAR_PRESOLVE_TOL_S Absolute zero tolerance employed for s_i in the presolve.
65	MSK_DPAR_UPPER_OBJ_CUT

continued on next page

continued from previous page

	If either a primal or dual feasible solution is found proving that the optimal objective value is outside, [MSK_DPAR_LOWER_OBJ_CUT, MSK_DPAR_UPPER_OBJ_CUT], then MOSEK is terminated.
16	MSK_DPAR_INTPNT_CO_TOL_DFEAS Dual feasibility tolerance used by the conic interior-point optimizer.
8	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.
49	MSK_DPAR_MIO_TOL_ABS_GAP Absolute optimality tolerance employed by the mixed-integer optimizer.
66	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 ∞.
50	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.
56	MSK_DPAR_NONCONVEX_TOL_OPT Optimality tolerance used by the nonconvex optimizer.
55	MSK_DPAR_NONCONVEX_TOL_FEAS Feasibility tolerance used by the nonconvex optimizer.
64	MSK_DPAR_SIMPLEX_ABS_TOL_PIV Absolute pivot tolerance employed by the simplex optimizers.
42	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.
5	MSK_DPAR_CHECK_CONVEXITY_REL_TOL This parameter controls when the full convexity check declares a problem to be non-convex. Increasing this tolerance relaxes the criteria for declaring the problem non-convex. A problem is declared non-convex if negative (positive) pivot elements are detected in the cholesky factor of a matrix which is required to be PSD (NSD). This parameter controles how much this non-negativity requirement may be violated. If d_i is the pivot element for column i, then the matrix Q is considered to not be PSD if: $d_i \leq -\lvert Q_{ii}\rvert * \texttt{check\_convexity\_rel\_tol}$
61	MSK_DPAR_PRESOLVE_TOL_X

continued on next page

continued from previous page

	Absolute zero tolerance employed for x_j in the presolve.
24	MSK_DPAR_INTPNT_NL_TOL_MU_RED
	Relative complementarity gap tolerance.
46	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.
6	MSK_DPAR_DATA_TOL_AIJ
	Absolute zero tolerance for elements in A. If any value A_{ij} is smaller than this parameter in absolute terms MOSEK will treat the values as zero and generate a warning.
15	MSK_DPAR_FEASREPAIR_TOL
	Tolerance for constraint enforcing upper bound on sum of weighted violations in feasibility repair.
30	MSK_DPAR_INTPNT_TOL_DSAFE
	Controls the initial dual starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly.
51	MSK_DPAR_MIO_TOL_FEAS
	Feasibility tolerance for mixed integer solver. Any solution with maximum infeasibility below this value will be considered feasible.
31	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.
25	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.
57	MSK_DPAR_OPTIMIZER_MAX_TIME
	Maximum amount of time the optimizer is allowed to spent on the optimization. A negative number means infinity.
14	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.
0	MSK_DPAR_ANA_SOL_INFEAS_TOL
	If a constraint violates its bound with an amount larger than this value, the constraint name, index and violation will be printed by the solution analyzer.
47	MSK_DPAR_MIO_REL_ADD_CUT_LIMITED

continued on next page

continued from previous page

	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.
32	`MSK_DPAR_INTPNT_TOL_MU_RED` Relative complementarity gap tolerance.
18	`MSK_DPAR_INTPNT_CO_TOL_MU_RED` Relative complementarity gap tolerance feasibility tolerance used by the conic interior-point optimizer.
21	`MSK_DPAR_INTPNT_CO_TOL_REL_GAP` Relative gap termination tolerance used by the conic interior-point optimizer.
39	`MSK_DPAR_LOWER_OBJ_CUT` If either a primal or dual feasible solution is found proving that the optimal objective value is outside, the interval [`MSK_DPAR_LOWER_OBJ_CUT`, `MSK_DPAR_UPPER_OBJ_CUT`], then MOSEK is terminated.
41	`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 criteria are never checked.
37	`MSK_DPAR_INTPNT_TOL_REL_STEP` Relative step size to the boundary for linear and quadratic optimization problems.
54	`MSK_DPAR_MIO_TOL_X` Absolute solution tolerance used in mixed-integer optimizer.
11	`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.
59	`MSK_DPAR_PRESOLVE_TOL_LIN_DEP` Controls when a constraint is determined to be linearly dependent.
63	`MSK_DPAR_SIM_LU_TOL_REL_PIV`

continued on next page

continued from previous page

	Relative pivot tolerance employed when computing the LU factorization of the basis in the simplex optimizers and in the basis identification procedure. A value closer to 1.0 generally improves numerical stability but typically also implies an increase in the computational work.
12	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.
28	MSK_DPAR_INTPNT_NL_TOL_REL_STEP Relative step size to the boundary for general nonlinear optimization problems.
38	MSK_DPAR_INTPNT_TOL_STEP_SIZE If the step size falls below the value of this parameter, then the interior-point optimizer assumes that it is stalled. It it does not not make any progress.
34	MSK_DPAR_INTPNT_TOL_PFEAS Primal feasibility tolerance used for linear and quadratic optimization problems.
1	MSK_DPAR_BASIS_REL_TOL_S Maximum relative dual bound violation allowed in an optimal basic solution.
17	MSK_DPAR_INTPNT_CO_TOL_INFEAS 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.
48	MSK_DPAR_MIO_REL_GAP_CONST This value is used to compute the relative gap for the solution to an integer optimization problem.
58	MSK_DPAR_PRESOLVE_TOL_AIJ Absolute zero tolerance employed for a_{ij} in the presolve.
44	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.
33	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 may be worthwhile to increase this parameter.
22	MSK_DPAR_INTPNT_NL_MERIT_BAL Controls if the complementarity and infeasibility is converging to zero at about equal rates.
3	MSK_DPAR_BASIS_TOL_X Maximum absolute primal bound violation allowed in an optimal basic solution.

continued on next page

continued from previous page	
36	MSK_DPAR_INTPNT_TOL_REL_GAP Relative gap termination tolerance.
7	MSK_DPAR_DATA_TOL_AIJ_HUGE An element in A which is larger than this value in absolute size causes an error.
10	MSK_DPAR_DATA_TOL_BOUND_WRN If a bound value is larger than this value in absolute size, then a warning message is issued.
9	MSK_DPAR_DATA_TOL_BOUND_INF Any bound which in absolute value is greater than this parameter is considered infinite.
35	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 may be worthwhile to increase this value.
19	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 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.
26	MSK_DPAR_INTPNT_NL_TOL_PFEAS Primal feasibility tolerance used when a nonlinear model is solved.
23	MSK_DPAR_INTPNT_NL_TOL_DFEAS Dual feasibility tolerance used when a nonlinear model is solved.
52	MSK_DPAR_MIO_TOL_REL_GAP Relative optimality tolerance employed by the mixed-integer optimizer.
29	MSK_DPAR_INTPNT_TOL_DFEAS Dual feasibility tolerance used for linear and quadratic optimization problems.
45	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.
53	MSK_DPAR_MIO_TOL_REL_RELAX_INT
	continued on next page

continued from previous page

	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.
62	MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL This parameter determines when columns are dropped in incomplete cholesky factorization doing reformulation of quadratic problems.
13	MSK_DPAR_DATA_TOL_QIJ Absolute zero tolerance for elements in Q matrices.
27	MSK_DPAR_INTPNT_NL_TOL_REL_GAP Relative gap termination tolerance for nonlinear problems.
20	MSK_DPAR_INTPNT_CO_TOL_PFEAS Primal feasibility tolerance used by the conic interior-point optimizer.

I.15 Feasibility repair types

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.16 License feature

Value	Name Description
2	MSK_FEATURE_PTOM Mixed-integer extension.
1	MSK_FEATURE_PTON Nonlinear extension.
0	MSK_FEATURE_PTS Base system.
3	MSK_FEATURE_PTOX Non-convex extension.

I.17 Integer information items.

Value	Name Description
57	MSK_IINF_RD_NUMINTVAR Number of integer-constrained variables read.
90	MSK_IINF_SOL_BAS_SOLSTA Solution status of the basic solution. Updated after each optimization.
97	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.
48	MSK_IINF_MIO_TOTAL_NUM_OBJ_CUTS Number of obj cuts.
88	MSK_IINF_SIM_SOLVE_DUAL Is non-zero if dual problem is solved.
30	MSK_IINF_MIO_NUMCON Number of constraints in the problem solved be the mixed-integer optimizer.
53	MSK_IINF_OPT_NUMVAR Number of variables in the problem solved when the optimizer is called
77	MSK_IINF_SIM_NUMVAR Number of variables in the problem solved by the simplex optimizer.
46	MSK_IINF_MIO_TOTAL_NUM_LATTICE_CUTS Number of lattice cuts.
71	MSK_IINF_SIM_NETWORK_PRIMAL_DEG_ITER The number of primal network degenerate iterations.
58	MSK_IINF_RD_NUMQ Number of nonempty Q matrices read.
0	MSK_IINF_ANA_PRO_NUM_CON Number of constraints in the problem.
19	MSK_IINF_INTPNT_FACTOR_NUM_OFFCOL Number of columns in the constraint matrix (or Jacobian) that has an offending structure.
11	MSK_IINF_ANA_PRO_NUM_VAR_INT Number of general integer variables.
69	MSK_IINF_SIM_NETWORK_DUAL_INF_ITER The number of iterations taken with dual infeasibility in the network optimizer.
8	MSK_IINF_ANA_PRO_NUM_VAR_CONT Number of continuous variables.
65	MSK_IINF_SIM_DUAL_ITER Number of dual simplex iterations during the last optimization.
61	MSK_IINF_SIM_DUAL_DEG_ITER

continued on next page

continued from previous page

	The number of dual degenerate iterations.
20	MSK_IINF_INTPNT_ITER
	Number of interior-point iterations since invoking the interior-point optimizer.
45	MSK_IINF_MIO_TOTAL_NUM_KNAPSUR_COVER_CUTS
	Number of knapsack cover cuts.
33	MSK_IINF_MIO_TOTAL_NUM_BASIS_CUTS
	Number of basis cuts.
76	MSK_IINF_SIM_NUMCON
	Number of constraints in the problem solved by the simplex optimizer.
83	MSK_IINF_SIM_PRIMAL_DUAL_ITER
	Number of primal dual simplex iterations during the last optimization.
5	MSK_IINF_ANA_PRO_NUM_CON_UP
	Number of constraints with an upper bound and an infinite lower bound.
36	MSK_IINF_MIO_TOTAL_NUM_CLIQUE_CUTS
	Number of clique cuts.
80	MSK_IINF_SIM_PRIMAL_DUAL_HOTSTART
	If 1 then the primal dual simplex algorithm is solving from an advanced basis.
22	MSK_IINF_INTPNT_SOLVE_DUAL
	Non-zero if the interior-point optimizer is solving the dual problem.
67	MSK_IINF_SIM_NETWORK_DUAL_HOTSTART
	If 1 then the dual network simplex algorithm is solving from an advanced basis.
54	MSK_IINF_OPTIMIZE_RESPONSE
	The reponse code returned by optimize.
93	MSK_IINF_SOL_ITR_PROSTA
	Problem status of the interior-point solution. Updated after each optimization.
60	MSK_IINF_RD_PROTYPE
	Problem type.
94	MSK_IINF_SOL_ITR_SOLSTA
	Solution status of the interior-point solution. Updated after each optimization.
2	MSK_IINF_ANA_PRO_NUM_CON_FR
	Number of unbounded constraints.
81	MSK_IINF_SIM_PRIMAL_DUAL_HOTSTART_LU
	If 1 then a valid basis factorization of full rank was located and used by the primal dual simplex algorithm.
31	MSK_IINF_MIO_NUMINT
	Number of integer variables in the problem solved be the mixed-integer optimizer.

continued on next page

continued from previous page

35	MSK_IINF_MIO_TOTAL_NUM_CARDGUB_CUTS Number of cardgub cuts.
38	MSK_IINF_MIO_TOTAL_NUM_CONTRA_CUTS Number of contra cuts.
49	MSK_IINF_MIO_TOTAL_NUM_PLAN_LOC_CUTS Number of loc cuts.
64	MSK_IINF_SIM_DUAL_INF_ITER The number of iterations taken with dual infeasibility.
32	MSK_IINF_MIO_NUMVAR Number of variables in the problem solved be the mixed-integer optimizer.
27	MSK_IINF_MIO_NUM_CUTS Number of cuts generated by the mixed-integer optimizer.
23	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.
6	MSK_IINF_ANA_PRO_NUM_VAR Number of variables in the problem.
95	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.
91	MSK_IINF_SOL_INT_PROSTA Problem status of the integer solution. Updated after each optimization.
66	MSK_IINF_SIM_NETWORK_DUAL_DEG_ITER The number of dual network degenerate iterations.
92	MSK_IINF_SOL_INT_SOLSTA Solution status of the integer solution. Updated after each optimization.
15	MSK_IINF_CACHE_SIZE_L1 L1 cache size used.
16	MSK_IINF_CACHE_SIZE_L2 L2 cache size used.
59	MSK_IINF_RD_NUMVAR Number of variables read.
79	MSK_IINF_SIM_PRIMAL_DUAL_DEG_ITER The number of degenerate major iterations taken by the primal dual simplex algorithm.
12	MSK_IINF_ANA_PRO_NUM_VAR_LO Number of variables with a lower bound and an infinite upper bound.
47	MSK_IINF_MIO_TOTAL_NUM_LIFT_CUTS Number of lift cuts.

continued on next page

continued from previous page

89	MSK_IINF_SOL_BAS_PROSTA Problem status of the basic solution. Updated after each optimization.
75	MSK_IINF_SIM_NETWORK_PRIMAL_ITER Number of primal network simplex iterations during the last optimization.
3	MSK_IINF_ANA_PRO_NUM_CON_LO Number of constraints with a lower bound and an infinite upper bound.
44	MSK_IINF_MIO_TOTAL_NUM_GUB_COVER_CUTS Number of GUB cover cuts.
68	MSK_IINF_SIM_NETWORK_DUAL_HOTSTART_LU If 1 then a valid basis factorization of full rank was located and used by the dual network simplex algorithm.
74	MSK_IINF_SIM_NETWORK_PRIMAL_INF_ITER The number of iterations taken with primal infeasibility in the network optimizer.
84	MSK_IINF_SIM_PRIMAL_HOTSTART If 1 then the primal simplex algorithm is solving from an advanced basis.
26	MSK_IINF_MIO_NUM_BRANCH Number of branches performed during the optimization.
96	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.
29	MSK_IINF_MIO_NUM_RELAX Number of relaxations solved during the optimization.
34	MSK_IINF_MIO_TOTAL_NUM_BRANCH Number of branches performed during the optimization.
42	MSK_IINF_MIO_TOTAL_NUM_GCD_CUTS Number of gcd cuts.
41	MSK_IINF_MIO_TOTAL_NUM_FLOW_COVER_CUTS Number of flow cover cuts.
28	MSK_IINF_MIO_NUM_INT_SOLUTIONS Number of integer feasible solutions that has been found.
85	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.
18	MSK_IINF_CPU_TYPE The type of cpu detected.
1	MSK_IINF_ANA_PRO_NUM_CON_EQ Number of equality constraints.
13	MSK_IINF_ANA_PRO_NUM_VAR_RA Number of variables with finite lower and upper bounds.

continued on next page

continued from previous page

86	MSK_IINF_SIM_PRIMAL_INF_ITER The number of iterations taken with primal infeasibility.
55	MSK_IINF_RD_NUMCON Number of constraints read.
56	MSK_IINF_RD_NUMCONE Number of conic constraints read.
10	MSK_IINF_ANA_PRO_NUM_VAR_FR Number of free variables.
25	MSK_IINF_MIO_NUM_ACTIVE_NODES Number of active nodes in the branch and bound tree.
50	MSK_IINF_MIO_TOTAL_NUM_RELAX Number of relaxations solved during the optimization.
7	MSK_IINF_ANA_PRO_NUM_VAR_BIN Number of binary (0-1) variables.
73	MSK_IINF_SIM_NETWORK_PRIMAL_HOTSTART_LU If 1 then a valid basis factorization of full rank was located and used by the primal network simplex algorithm.
87	MSK_IINF_SIM_PRIMAL_ITER Number of primal simplex iterations during the last optimization.
62	MSK_IINF_SIM_DUAL_HOTSTART If 1 then the dual simplex algorithm is solving from an advanced basis.
24	MSK_IINF_MIO_INITIAL_SOLUTION Is non-zero if an initial integer solution is specified.
21	MSK_IINF_INTPNT_NUM_THREADS Number of threads that the interior-point optimizer is using.
63	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.
14	MSK_IINF_ANA_PRO_NUM_VAR_UP Number of variables with an upper bound and an infinite lower bound. This value is set by
70	MSK_IINF_SIM_NETWORK_DUAL_ITER Number of dual network simplex iterations during the last optimization.
9	MSK_IINF_ANA_PRO_NUM_VAR_EQ Number of fixed variables.
17	MSK_IINF_CONCURRENT_FASTEST_OPTIMIZER The type of the optimizer that finished first in a concurrent optimization.
51	MSK_IINF_MIO_USER_OBJ_CUT If it is non-zero, then the objective cut is used.
43	MSK_IINF_MIO_TOTAL_NUM_GOMORY_CUTS Number of Gomory cuts.
72	MSK_IINF_SIM_NETWORK_PRIMAL_HOTSTART

continued on next page

continued from previous page

	If 1 then the primal network simplex algorithm is solving from an advanced basis.
40	MSK_IINF_MIO_TOTAL_NUM_DISAGG_CUTS Number of diasagg cuts.
37	MSK_IINF_MIO_TOTAL_NUM_COEF_REDC_CUTS Number of coef. redc. cuts.
82	MSK_IINF_SIM_PRIMAL_DUAL_INF_ITER The number of master iterations with dual infeasibility taken by the primal dual simplex algorithm.
4	MSK_IINF_ANA_PRO_NUM_CON_RA Number of constraints with finite lower and upper bounds.
39	MSK_IINF_MIO_TOTAL_NUM_CUTS Total number of cuts generated by the mixed-integer optimizer.
52	MSK_IINF_OPT_NUMCON Number of constraints in the problem solved when the optimizer is called.
78	MSK_IINF_SIM_PRIMAL_DEG_ITER The number of primal degenerate iterations.

I.18 Information item types

Value	Name Description
0	MSK_INF_DOU_TYPE Is a double information type.
2	MSK_INF_LINT_TYPE Is a long integer.
1	MSK_INF_INT_TYPE Is an integer.

I.19 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.20 Integer parameters

Value	Name Description
175	MSK_IPAR_SIM_STABILITY_PRIORITY Controls how high priority the numerical stability should be given.
125	MSK_IPAR_READ_ADD_CONE Additional number of conic constraints that is made room for in the problem.
166	MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD An exprimental feature.
204	MSK_IPAR_WRITE_MPS_STRICT Controls whether the written MPS file satisfies the MPS format strictly or not.
25	MSK_IPAR_INFEAS_REPORT_AUTO Controls whether an infeasibility report is automatically produced after the optimization if the problem is primal or dual infeasible.
93	MSK_IPAR_MIO_NODE_OPTIMIZER Controls which optimizer is employed at the non-root nodes in the mixed-integer optimizer.
118	MSK_IPAR_PRESOLVE_LEVEL Currently not used.
127	MSK_IPAR_READ_ADD_VAR Additional number of variables that is made room for in the problem.
121	MSK_IPAR_PRESOLVE_USE Controls whether the presolve is applied to a problem before it is optimized.
70	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.
109	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.
186	MSK_IPAR_WRITE_BAS_HEAD Controls whether the header section is written to the basic solution file.
79	MSK_IPAR_MIO_BRANCH_PRIORITIES_USE Controls whether branching priorities are used by the mixed-integer optimizer.
83	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.
188	MSK_IPAR_WRITE_DATA_COMPRESSED

continued on next page

continued from previous page

	Controls whether the data file is compressed while it is written. 0 means no compression while higher values mean more compression.
140	`MSK_IPAR_READ_MPS_RELAX`
	If this option is turned on, then mixed integer constraints are ignored when a problem is read.
106	`MSK_IPAR_OPF_WRITE_PARAMETERS`
	Write a parameter section in an OPF file.
129	`MSK_IPAR_READ_CON`
	Expected maximum number of constraints to be read. The option is only used by fast MPS and LP file readers.
196	`MSK_IPAR_WRITE_INT_VARIABLES`
	Controls whether the variables section is written to the integer solution file.
123	`MSK_IPAR_READ_ADD_ANZ`
	Additional number of non-zeros in A that is made room for in the problem.
36	`MSK_IPAR_INTPNT_ORDER_METHOD`
	Controls the ordering strategy used by the interior-point optimizer when factorizing the Newton equation system.
110	`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.
69	`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.
143	`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.
59	`MSK_IPAR_LOG_INFEAS_ANA`
	Controls amount of output printed by the infeasibility analyzer procedures. A higher level implies that more information is logged.
168	`MSK_IPAR_SIM_PRIMAL_SELECTION`
	Controls the choice of the incoming variable, known as the selection strategy, in the primal simplex optimizer.
194	`MSK_IPAR_WRITE_INT_CONSTRAINTS`
	Controls whether the constraint section is written to the integer solution file.
199	`MSK_IPAR_WRITE_LP_STRICT_FORMAT`
	Controls whether LP output files satisfy the LP format strictly.
148	`MSK_IPAR_SENSITIVITY_TYPE`
	Controls which type of sensitivity analysis is to be performed.
153	`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.
62	`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.
108	`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.
14	`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.
99	`MSK_IPAR_MIO_ROOT_OPTIMIZER` Controls which optimizer is employed at the root node in the mixed-integer optimizer.
191	`MSK_IPAR_WRITE_FREE_CON` Controls whether the free constraints are written to the data file.
115	`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.
101	`MSK_IPAR_NONCONVEX_MAX_ITERATIONS` Maximum number of iterations that can be used by the nonconvex optimizer.
88	`MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER` Controls the size of the local search space when doing local branching.
192	`MSK_IPAR_WRITE_GENERIC_NAMES` Controls whether the generic names or user-defined names are used in the data file.
184	`MSK_IPAR_WARNING_LEVEL` Warning level.
51	`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.
16	`MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX`

continued on next page

continued from previous page

	Priority of the dual simplex algorithm when selecting solvers for concurrent optimization.
67	MSK_IPAR_LOG_PRESOLVE Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.
126	MSK_IPAR_READ_ADD_QNZ Additional number of non-zeros in the Q matrices that is made room for in the problem.
206	MSK_IPAR_WRITE_SOL_CONSTRAINTS Controls whether the constraint section is written to the solution file.
35	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.
128	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.
92	MSK_IPAR_MIO_MODE Controls whether the optimizer includes the integer restrictions when solving a (mixed) integer optimization problem.
134	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.
71	MSK_IPAR_LOG_SIM Controls amount of output printed by the simplex optimizer. A higher level implies that more information is logged.
41	MSK_IPAR_LIC_TRH_EXPIRY_WRN If a license feature expires in a numbers days less than the value of this parameter then a warning will be issued.
182	MSK_IPAR_SOLUTION_CALLBACK Indicates whether solution call-backs will be performed during the optimization.
173	MSK_IPAR_SIM_SCALING_METHOD Controls how the problem is scaled before a simplex optimizer is used.
72	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.
63	MSK_IPAR_LOG_NONCONVEX Controls amount of output printed by the nonconvex optimizer.
22	MSK_IPAR_FEASREPAIR_OPTIMIZE Controls which type of feasibility analysis is to be performed.

continued on next page

continued from previous page

198	`MSK_IPAR_WRITE_LP_QUOTED_NAMES` If this option is turned on, then MOSEK will quote invalid LP names when writing an LP file.
55	`MSK_IPAR_LOG_FACTOR` If turned on, then the factor log lines are added to the log.
4	`MSK_IPAR_AUTO_UPDATE_SOL_INFO` Controls whether the solution information items are automatically updated after an optimization is performed.
203	`MSK_IPAR_WRITE_MPS_QUOTED_NAMES` If a name contains spaces (blanks) when writing an MPS file, then the quotes will be removed.
141	`MSK_IPAR_READ_MPS_WIDTH` Controls the maximal number of characters allowed in one line of the MPS file.
183	`MSK_IPAR_TIMING_LEVEL` Controls the a amount of timing performed inside MOSEK.
65	`MSK_IPAR_LOG_ORDER` If turned on, then factor lines are added to the log.
82	`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
5	`MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE` If a slack variable is in the basis, then the corresponding column in the basis is a unit vector with -1 in the right position. However, if this parameter is set to `MSK_ON`, -1 is replaced by 1.
8	`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`.
94	`MSK_IPAR_MIO_NODE_SELECTION`

continued on next page

continued from previous page

	Controls the node selection strategy employed by the mixed-integer optimizer.
2	`MSK_IPAR_ANA_SOL_PRINT_VIOLATED`
	Controls whether a list of violated constraints is printed.
181	`MSK_IPAR_SOL_READ_WIDTH`
	Controls the maximal acceptable width of line in the solutions when read by MOSEK.
120	`MSK_IPAR_PRESOLVE_LINDEP_WORK_LIM`
	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.
33	`MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS`
	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.
124	`MSK_IPAR_READ_ADD_CON`
	Additional number of constraints that is made room for in the problem.
53	`MSK_IPAR_LOG_CONCURRENT`
	Controls amount of output printed by the concurrent optimizer.
73	`MSK_IPAR_LOG_SIM_MINOR`
	Currently not in use.
159	`MSK_IPAR_SIM_MAX_ITERATIONS`
	Maximum number of iterations that can be used by a simplex optimizer.
31	`MSK_IPAR_INTPNT_MAX_ITERATIONS`
	Controls the maximum number of iterations allowed in the interior-point optimizer.
20	`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.
50	`MSK_IPAR_LOG_BI`
	Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.
32	`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.
197	`MSK_IPAR_WRITE_LP_LINE_WIDTH`
	Maximum width of line in an LP file written by MOSEK.
180	`MSK_IPAR_SOL_READ_NAME_WIDTH`

continued on next page

continued from previous page

	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.
45	`MSK_IPAR_LICENSE_DEBUG` This option is used to turn on debugging of the incense manager.
48	`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.
1	`MSK_IPAR_ANA_SOL_BASIS` Controls whether the basis matrix is analyzed in solaution analyzer.
116	`MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES` Control the maximum number of times the eliminator is tried.
193	`MSK_IPAR_WRITE_GENERIC_NAMES_IO` Index origin used in generic names.
15	`MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS` The maximum number of simultaneous optimizations that will be started by the concurrent optimizer.
169	`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.
154	`MSK_IPAR_SIM_DUAL_SELECTION` Controls the choice of the incoming variable, known as the selection strategy, in the dual simplex optimizer.
174	`MSK_IPAR_SIM_SOLVE_FORM` Controls whether the primal or the dual problem is solved by the primal-/dual- simplex optimizer.
13	`MSK_IPAR_CHECK_CONVEXITY` Specify the level of convexity check on quadratic problems
122	`MSK_IPAR_QO_SEPARABLE_REFORMULATION` Determine if Quadratic programing problems should be reformulated to separable form.
76	`MSK_IPAR_LP_WRITE_IGNORE_INCOMPATIBLE_ITEMS` Controls the result of writing a problem containing incompatible items to an LP file.
144	`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.
95	`MSK_IPAR_MIO_OPTIMIZER_MODE` An exprimental feature.
60	`MSK_IPAR_LOG_INTPNT` Controls amount of output printed printed by the interior-point optimizer. A higher level implies that more information is logged.

continued on next page

continued from previous page

61	MSK_IPAR_LOG_MIO Controls the log level for the mixed-integer optimizer. A higher level implies that more information is logged.
156	MSK_IPAR_SIM_HOTSTART Controls the type of hot-start that the simplex optimizer perform.
66	MSK_IPAR_LOG_PARAM Controls the amount of information printed out about parameter changes.
189	MSK_IPAR_WRITE_DATA_FORMAT Controls the file format when writing task data to a file.
155	MSK_IPAR_SIM_EXPLOIT_DUPVEC Controls if the simplex optimizers are allowed to exploit duplicated columns.
78	MSK_IPAR_MIO_BRANCH_DIR Controls whether the mixed-integer optimizer is branching up or down by default.
29	MSK_IPAR_INTPNT_FACTOR_DEBUG_LVL Controls factorization debug level.
179	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.
46	MSK_IPAR_LICENSE_PAUSE_TIME If MSK_IPAR_LICENSE_WAIT=MSK_ON and no license is available, then MOSEK sleeps a number of milliseconds between each check of whether a license has become free.
96	MSK_IPAR_MIO_PRESOLVE_AGGREGATE Controls whether the presolve used by the mixed-integer optimizer tries to aggregate the constraints.
209	MSK_IPAR_WRITE_TASK_INC_SOL Controls whether the solutions are stored in the task file too.
43	MSK_IPAR_LICENSE_CACHE_TIME Setting this parameter no longer has any effect. Please see MSK_IPAR_CACHE_LICENSE for an alternative.
9	MSK_IPAR_BI_MAX_ITERATIONS Controls the maximum number of simplex iterations allowed to optimize a basis after the basis identification.
157	MSK_IPAR_SIM_HOTSTART_LU Determines if the simplex optimizer should exploit the initial factorization.
111	MSK_IPAR_OPF_WRITE_SOLUTIONS Enable inclusion of solutions in the OPF files.
162	MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART

continued on next page

continued from previous page

	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.
119	`MSK_IPAR_PRESOLVE_LINDEP_USE` Controls whether the linear constraints are checked for linear dependencies.
114	`MSK_IPAR_PARAM_READ_IGN_ERROR` If turned on, then errors in paramter settings is ignored.
104	`MSK_IPAR_OPF_WRITE_HEADER` Write a text header with date and MOSEK version in an OPF file.
81	`MSK_IPAR_MIO_CONT_SOL` Controls the meaning of the interior-point and basic solutions in mixed integer problems.
102	`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.
195	`MSK_IPAR_WRITE_INT_HEAD` Controls whether the header section is written to the integer solution file.
40	`MSK_IPAR_INTPNT_STARTING_POINT` Starting point used by the interior-point optimizer.
49	`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.
19	`MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX` Priority of the primal simplex algorithm when selecting solvers for concurrent optimization.
138	`MSK_IPAR_READ_MPS_OBJ_SENSE` If turned on, the MPS reader uses the objective sense section. Otherwise the MPS reader ignores it.
10	`MSK_IPAR_CACHE_LICENSE`

continued on next page

continued from previous page

	Specifies if the license is kept checked out for the lifetime of the mosek environment (on) or returned to the server immediately after the optimization (off). Check-in and check-out of licenses have an overhead. Frequent communication with the license server should be avoided.
74	`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`.
28	`MSK_IPAR_INTPNT_DIFF_STEP` Controls whether different step sizes are allowed in the primal and dual space.
172	`MSK_IPAR_SIM_SCALING` Controls how much effort is used in scaling the problem before a simplex optimizer is used.
200	`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.
146	`MSK_IPAR_SENSITIVITY_ALL` Not applicable.
178	`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.
7	`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`.
56	`MSK_IPAR_LOG_FEASREPAIR` Controls the amount of output printed when performing feasibility repair.
39	`MSK_IPAR_INTPNT_SOLVE_FORM` Controls whether the primal or the dual problem is solved.
103	`MSK_IPAR_OPF_MAX_TERMS_PER_LINE` The maximum number of terms (linear and quadratic) per line when an OPF file is written.
205	`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.
149	`MSK_IPAR_SIM_BASIS_FACTOR_USE`

continued on next page

continued from previous page

	Controls whether a (LU) factorization of the basis is used in a hot-start. Forcing a refactorization sometimes improves the stability of the simplex optimizers, but in most cases there is a performance penanlty.
210	MSK_IPAR_WRITE_XML_MODE Controls if linear coefficients should be written by row or column when writing in the XML file format.
37	MSK_IPAR_INTPNT_REGULARIZATION_USE Controls whether regularization is allowed.
6	MSK_IPAR_BI_CLEAN_OPTIMIZER Controls which simplex optimizer is used in the clean-up phase.
97	MSK_IPAR_MIO_PRESOLVE_PROBING Controls whether the mixed-integer presolve performs probing. Probing can be very time consuming.
42	MSK_IPAR_LICENSE_ALLOW_OVERUSE Controls if license overuse is allowed when caching licenses
24	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.
187	MSK_IPAR_WRITE_BAS_VARIABLES Controls whether the variables section is written to the basic solution file.
75	MSK_IPAR_LOG_STORAGE When turned on, MOSEK prints messages regarding the storage usage and allocation.
98	MSK_IPAR_MIO_PRESOLVE_USE Controls whether presolve is performed by the mixed-integer optimizer.
135	MSK_IPAR_READ_LP_QUOTED_NAMES If a name is in quotes when reading an LP file, the quotes will be removed.
27	MSK_IPAR_INTPNT_BASIS Controls whether the interior-point optimizer also computes an optimal basis.
54	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.
137	MSK_IPAR_READ_MPS_KEEP_INT Controls whether MOSEK should keep the integer restrictions on the variables while reading the MPS file.
91	MSK_IPAR_MIO_MAX_NUM_SOLUTIONS

continued on next page

continued from previous page

	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.
44	`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.
208	`MSK_IPAR_WRITE_SOL_VARIABLES` Controls whether the variables section is written to the solution file.
147	`MSK_IPAR_SENSITIVITY_OPTIMIZER` Controls which optimizer is used for optimal partition sensitivity analysis.
201	`MSK_IPAR_WRITE_MPS_INT` Controls if marker records are written to the MPS file to indicate whether variables are integer restricted.
160	`MSK_IPAR_SIM_MAX_NUM_SETBACKS` Controls how many set-backs are allowed within a simplex optimizer. A set-back is an event where the optimizer moves in the wrong direction. This is impossible in theory but may happen due to numerical problems.
21	`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.
17	`MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX` Priority of the free simplex optimizer when selecting solvers for concurrent optimization.
133	`MSK_IPAR_READ_KEEP_FREE_CON` Controls whether the free constraints are included in the problem.
57	`MSK_IPAR_LOG_FILE` If turned on, then some log info is printed when a file is written or read.
18	`MSK_IPAR_CONCURRENT_PRIORITY_INTPNT` Priority of the interior-point algorithm when selecting solvers for concurrent optimization.
164	`MSK_IPAR_SIM_NON_SINGULAR` Controls if the simplex optimizer ensures a non-singular basis, if possible.
190	`MSK_IPAR_WRITE_DATA_PARAM` If this option is turned on the parameter settings are written to the data file as parameters.
150	`MSK_IPAR_SIM_DEGEN` Controls how aggressively degeneration is handled.

continued on next page

continued from previous page

105	MSK_IPAR_OPF_WRITE_HINTS Write a hint section with problem dimensions in the beginning of an OPF file.
117	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.
86	MSK_IPAR_MIO_HOTSTART Controls whether the integer optimizer is hot-started.
136	MSK_IPAR_READ_MPS_FORMAT Controls how strictly the MPS file reader interprets the MPS format.
113	MSK_IPAR_PARAM_READ_CASE_NAME If turned on, then names in the parameter file are case sensitive.
139	MSK_IPAR_READ_MPS_QUOTED_NAMES If a name is in quotes when reading an MPS file, then the quotes will be removed.
64	MSK_IPAR_LOG_OPTIMIZER Controls the amount of general optimizer information that is logged.
202	MSK_IPAR_WRITE_MPS_OBJ_SENSE If turned off, the objective sense section is not written to the MPS file.
34	MSK_IPAR_INTPNT_NUM_THREADS Controls the number of threads employed by the interior-point optimizer. If set to a positive number MOSEK will use this number of threads. If zero the number of threads used will equal the number of cores detected on the machine.
89	MSK_IPAR_MIO_MAX_NUM_BRANCHES Maximum number of branches allowed during the branch and bound search. A negative value means infinite.
165	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.
80	MSK_IPAR_MIO_CONSTRUCT_SOL If set to MSK_ON and all integer variables have been given a value for which a feasible mixed integer solution exists, then MOSEK generates an initial solution to the mixed integer problem by fixing all integer values and solving the remaining problem.
3	MSK_IPAR_AUTO_SORT_A_BEFORE_OPT

continued on next page

continued from previous page

	Controls whether the elements in each column of A are sorted before an optimization is performed. This is not required but makes the optimization more deterministic.
100	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.
152	MSK_IPAR_SIM_DUAL_PHASEONE_METHOD
	An exprimental feature.
158	MSK_IPAR_SIM_INTEGER
	An exprimental feature.
167	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.
130	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.
112	MSK_IPAR_OPTIMIZER
	The paramter controls which optimizer is used to optimize the task.
77	MSK_IPAR_MAX_NUM_WARNINGS
	Waning level. A higher value results in more warnings.
47	MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS
	Controls whether license features expire warnings are suppressed.
207	MSK_IPAR_WRITE_SOL_HEAD
	Controls whether the header section is written to the solution file.
185	MSK_IPAR_WRITE_BAS_CONSTRAINTS
	Controls whether the constraint section is written to the basic solution file.
84	MSK_IPAR_MIO_FEASPUMP_LEVEL
	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.
23	MSK_IPAR_INFEAS_GENERIC_NAMES

continued on next page

continued from previous page

	Controls whether generic names are used when an infeasible subproblem is created.
161	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.
68	MSK_IPAR_LOG_RESPONSE Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.
26	MSK_IPAR_INFEAS_REPORT_LEVEL Controls the amount of information presented in an infeasibility report. Higher values imply more information.
11	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.
12	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.
176	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).
131	MSK_IPAR_READ_DATA_COMPRESSED If this option is turned on,it is assumed that the data file is compressed.
142	MSK_IPAR_READ_Q_MODE Controls how the Q matrices are read from the MPS file.
107	MSK_IPAR_OPF_WRITE_PROBLEM Write objective, constraints, bounds etc. to an OPF file.
52	MSK_IPAR_LOG_CHECK_CONVEXITY

continued on next page

continued from previous page

	Controls logging in convexity check on quadratic problems. Set to a positive value to turn logging on. If a quadratic coefficient matrix is found to violate the requirement of PSD (NSD) then a list of negative (positive) pivot elements is printed. The absolute value of the pivot elements is also shown.
132	`MSK_IPAR_READ_DATA_FORMAT` Format of the data file to be read.
151	`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.
163	`MSK_IPAR_SIM_NETWORK_DETECT_METHOD` Controls which type of detection method the network extraction should use.
145	`MSK_IPAR_READ_VAR` Expected maximum number of variable to be read. The option is used only by MPS and LP file readers.
58	`MSK_IPAR_LOG_HEAD` If turned on, then a header line is added to the log.
170	`MSK_IPAR_SIM_REFORMULATION` Controls if the simplex optimizers are allowed to reformulate the problem.
171	`MSK_IPAR_SIM_SAVE_LU` Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.
30	`MSK_IPAR_INTPNT_FACTOR_METHOD` Controls the method used to factor the Newton equation system.
90	`MSK_IPAR_MIO_MAX_NUM_RELAXS` Maximum number of relaxations allowed during the branch and bound search. A negative value means infinite.
177	`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.
85	`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.
87	`MSK_IPAR_MIO_KEEP_BASIS` Controls whether the integer presolve keeps bases in memory. This speeds on the solution process at cost of bigger memory consumption.

continued on next page

continued from previous page

38	MSK_IPAR_INTPNT_SCALING Controls how the problem is scaled before the interior-point optimizer is used.

I.21 Language selection constants

Value	Name Description
1	MSK_LANG_DAN Danish language selection
0	MSK_LANG_ENG English language selection

I.22 Long integer information items.

Value	Name Description
6	MSK_LIINF_BI_CLEAN_PRIMAL_ITER Number of primal clean iterations performed in the basis identification.
9	MSK_LIINF_INTPNT_FACTOR_NUM_NZ Number of non-zeros in factorization.
10	MSK_LIINF_MIO_INTPNT_ITER Number of interior-point iterations performed by the mixed-integer optimizer.
4	MSK_LIINF_BI_CLEAN_PRIMAL_DUAL_ITER Number of primal-dual clean iterations performed in the basis identification.
3	MSK_LIINF_BI_CLEAN_PRIMAL_DUAL_DEG_ITER Number of primal-dual degenerate clean iterations performed in the basis identification.
2	MSK_LIINF_BI_CLEAN_PRIMAL_DEG_ITER Number of primal degenerate clean iterations performed in the basis identification.
1	MSK_LIINF_BI_CLEAN_DUAL_ITER Number of dual clean iterations performed in the basis identification.
13	MSK_LIINF_RD_NUMQNZ Number of Q non-zeros.
12	MSK_LIINF_RD_NUMANZ Number of non-zeros in A that is read.
8	MSK_LIINF_BI_PRIMAL_ITER

continued on next page

continued from previous page

Value	Name / Description
	Number of primal pivots performed in the basis identification.
7	MSK_LIINF_BI_DUAL_ITER Number of dual pivots performed in the basis identification.
0	MSK_LIINF_BI_CLEAN_DUAL_DEG_ITER Number of dual degenerate clean iterations performed in the basis identification.
11	MSK_LIINF_MIO_SIMPLEX_ITER Number of simplex iterations performed by the mixed-integer optimizer.
5	MSK_LIINF_BI_CLEAN_PRIMAL_DUAL_SUB_ITER Number of primal-dual subproblem clean iterations performed in the basis identification.

I.23 Mark

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.24 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

continued on next page

continued from previous page

	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.25 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.26 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.27 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.28 Message keys

Value	Name Description
1000	MSK_MSG_READING_FILE None
1001	MSK_MSG_WRITING_FILE None
1100	MSK_MSG_MPS_SELECTED None

I.29 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.30 Objective sense types

Value	Name Description
1	MSK_OBJECTIVE_SENSE_MINIMIZE The problem should be minimized.

continued on next page

continued from previous page	
0	MSK_OBJECTIVE_SENSE_UNDEFINED The objective sense is undefined.
2	MSK_OBJECTIVE_SENSE_MAXIMIZE The problem should be maximized.

I.31 On/off

Value	Name Description
1	MSK_ON Switch the option on.
0	MSK_OFF Switch the option off.

I.32 Optimizer types

Value	Name Description
1	MSK_OPTIMIZER_INTPNT The interior-point optimizer is used.
10	MSK_OPTIMIZER_CONCURRENT The optimizer for nonconvex nonlinear problems.
8	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.
6	MSK_OPTIMIZER_PRIMAL_DUAL_SIMPLEX The primal dual simplex optimizer is used.
2	MSK_OPTIMIZER_CONIC The optimizer for problems having conic constraints.
9	MSK_OPTIMIZER_NONCONVEX The optimizer for nonconvex nonlinear problems.
3	MSK_OPTIMIZER_QCONE For internal use only.
4	MSK_OPTIMIZER_PRIMAL_SIMPLEX The primal simplex optimizer is used.
7	MSK_OPTIMIZER_FREE_SIMPLEX One of the simplex optimizers is used.

I.33 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 The ordering method is chosen automatically.
3	MSK_ORDER_METHOD_GRAPHPAR1 Graph partitioning based ordering.

I.34 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.35 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.36 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.37 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.38 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`

continued on next page

continued from previous page	
	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 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.39 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.40 Response codes

Value	Name Description
352	MSK_RES_WRN_SOL_FILE_IGNORED_VAR One or more lines in the variable section were ignored when reading a solution file.
1218	MSK_RES_ERR_PARAM_TYPE
	continued on next page

continued from previous page

	The parameter type is invalid.
1203	MSK_RES_ERR_INDEX_IS_TOO_SMALL An index in an argument is too small.
2501	MSK_RES_ERR_INV_MARKI Invalid value in marki.
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.
1268	MSK_RES_ERR_INV_SKX Invalid value in `skx`.
1551	MSK_RES_ERR_MIO_NO_OPTIMIZER No optimizer is available for the current class of integer optimization problems.
4009	MSK_RES_TRM_MIO_NUM_BRANCHES The mixed-integer optimizer terminated as to the maximum number of branches was reached.
4004	MSK_RES_TRM_MIO_NEAR_ABS_GAP The mixed-integer optimizer terminated because the near optimal absolute gap tolerance was satisfied.
2001	MSK_RES_ERR_NO_DUAL_INFEAS_CER A certificate of infeasibility is not available.
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

continued on next page

continued from previous page

	MOSEK has the following solution types: • an interior-point solution, • an basic solution, • and an integer solution. Each optimizer may set one or more of these solutions; e.g by default a successful optimization with the interior-point optimizer defines the interior-point solution, and, for linear problems, also the basic solution. This error occurs when asking for a solution or for information about a solution that 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.
1234	`MSK_RES_ERR_INF_LINT_NAME` A long integer information name is invalid.
903	`MSK_RES_WRN_ANA_CLOSE_BOUNDS` This warning is issued by problem analyzer, if ranged constraints or variables with very close upper and lower bounds are detected. One should consider treating such constraints as equalities and such variables as constants.
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.
1350	`MSK_RES_ERR_SOL_FILE_INVALID_NUMBER` An invalid number is specified in a solution file.
2800	`MSK_RES_ERR_LU_MAX_NUM_TRIES` Could not compute the LU factors of the matrix within the maximum number of allowed tries.
1267	`MSK_RES_ERR_INV_SKC` Invalid value in `skc`.

continued on next page

continued from previous page

201	`MSK_RES_WRN_DROPPED_NZ_QOBJ` One or more non-zero elements were dropped in the Q matrix in the objective.
3000	`MSK_RES_ERR_INTERNAL` An internal error occurred. Please report this problem.
1610	`MSK_RES_ERR_BASIS_FACTOR` The factorization of the basis is invalid.
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.
1590	`MSK_RES_ERR_OVERFLOW` A computation produced an overflow i.e. a very large number.
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.
1225	`MSK_RES_ERR_INF_LINT_INDEX` A long integer information index is out of range for the specified type.
4008	`MSK_RES_TRM_MIO_NUM_RELAXS` The mixed-integer optimizer terminated as the maximum number of relaxations was reached.
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.
1106	`MSK_RES_ERR_MPS_UNDEF_VAR_NAME` An undefined variable name occurred in an MPS file.
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`

continued on next page

continued from previous page

	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`.
705	MSK_RES_WRN_ZEROS_IN_SPARSE_ROW One or more (near) zero elements are specified in a sparse row of a matrix. It is redundant to specify zero elements. Hence it may indicate an error.
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.
3102	MSK_RES_ERR_AD_INVALID_CODELIST The code list data was invalid.
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`.
1432	MSK_RES_ERR_USER_NLO_FUNC The user-defined nonlinear function reported an error.

continued on next page

continued from previous page

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`.
1431	MSK_RES_ERR_USER_FUNC_RET_DATA An user function returned invalid data.
3900	MSK_RES_ERR_SIZE_LICENSE_NUMCORES The computer contains more cpu cores than the license allows for.
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 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.
1002	MSK_RES_ERR_LICENSE_VERSION The license is valid for another version of MOSEK.
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

continued on next page

continued from previous page

	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.
1461	`MSK_RES_ERR_NAN_IN_BLC` l^c contains an invalid floating point value, i.e. a `NaN`.
3058	`MSK_RES_ERR_SEN_NUMERICAL` Numerical difficulties encountered performing the sensitivity analysis.
3052	`MSK_RES_ERR_SEN_INDEX_RANGE` Index out of range in the sensitivity analysis file.
1027	`MSK_RES_ERR_LICENSE_NO_SERVER_SUPPORT` The license server does not support the requested feature. Possible reasons for this error include: • The feature has expired. • The feature's start date is later than today's date. • The version requested is higher than feature's the highest supported version. • A corrupted license file. Try restarting the license and inspect the license server debug file, usually called `lmgrd.log`.
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.
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.
1172	`MSK_RES_ERR_OPF_PREMATURE_EOF` Premature end of file in an OPF file.
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`.

continued on next page

continued from previous page

1066	MSK_RES_ERR_LIVING_TASKS All tasks associated with an enviroment must be deleted before the environment is deleted. There are still some undeleted tasks.
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.
1171	MSK_RES_ERR_LP_INVALID_CON_NAME A constraint name is invalid when used in an LP formatted file.
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.
710	MSK_RES_WRN_ZEROS_IN_SPARSE_COL One or more (near) zero elements are specified in a sparse column of a matrix. It is redundant to specify zero elements. Hence, it may indicate an error.
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.
800	MSK_RES_WRN_INCOMPLETE_LINEAR_DEPENDENCY_CHECK The linear dependency check(s) was not completed and therefore the A matrix may contain linear dependencies.
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 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.
1048	MSK_RES_ERR_THREAD_CREATE

continued on next page

continued from previous page

	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.
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.
3059	MSK_RES_ERR_CONCURRENT_OPTIMIZER An unsupported optimizer was chosen for use with the concurrent optimizer.
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 an $a_{i,j}$ element in A. The parameter MSK_DPAR_DATA_TOL_AIJ_LARGE controls when an $a_{i,j}$ is considered large.
1208	MSK_RES_ERR_PARAM_NAME_STR The parameter name is not correct for a string parameter.

continued on next page

continued from previous page

1018	MSK_RES_ERR_LICENSE_FEATURE 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.
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.
1403	MSK_RES_ERR_INV_QOBJ_VAL Invalid value in `qoval`.
1400	MSK_RES_ERR_INFINITE_BOUND A numerically huge bound value is specified.
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.
1262	MSK_RES_ERR_LAST Invalid index `last`. A given index was out of expected range.
1151	MSK_RES_ERR_LP_EMPTY

continued on next page

continued from previous page

	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.
2505	MSK_RES_ERR_CANNOT_CLONE_NL A task with a nonlinear function call-back cannot be cloned.
2520	MSK_RES_ERR_INVALID_ACCMODE An invalid access mode is specified.
1250	MSK_RES_ERR_NUMCONLIM Maximum number of constraints limit is exceeded.
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.
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.
54	MSK_RES_WRN_LARGE_CON_FX An equality constraint is fixed to a numerically large value. This can cause numerical problems.
1280	MSK_RES_ERR_INV_NAME_ITEM An invalid name item code is used.
3106	MSK_RES_ERR_AD_MISSING_RETURN The code list data was invalid. Missing return operation in function.
53	MSK_RES_WRN_LARGE_UP_BOUND A numerically large upper bound value is specified.
3910	MSK_RES_ERR_INFEAS_UNDEFINED The requested value is not defined for this solution type.
901	MSK_RES_WRN_ANA_C_ZERO This warning is issued by the problem analyzer, if the coefficients in the linear part of the objective are all zero.
1112	MSK_RES_ERR_MPS_MUL_CON_NAME A constraint name was specified multiple times in the ROWS section.

continued on next page

continued from previous page

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 The optimizer terminated due to the return of the user-defined call-back function.
805	MSK_RES_WRN_CONSTRUCT_SOLUTION_INFEAS After fixing the integer variables at the suggested values then the problem is infeasible.
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 An integer information name is invalid.
1107	MSK_RES_ERR_MPS_INV_CON_KEY An invalid constraint key occurred in an MPS file.
1425	MSK_RES_ERR_FIXED_BOUND_VALUES A fixed constraint/variable has been specified using the bound keys but the numerical value of the lower and upper bound is different.
4025	MSK_RES_TRM_NUMERICAL_PROBLEM The optimizer terminated due to numerical problems.
3056	MSK_RES_ERR_SEN_INVALID_REGEXP Syntax error in regexp or regexp longer than 1024.
52	MSK_RES_WRN_LARGE_LO_BOUND A numerically large lower bound value is specified.
3999	MSK_RES_ERR_API_INTERNAL An internal fatal error occurred in an interface function.
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

continued on next page

continued from previous page

	The solution item y is undefined.
3200	MSK_RES_ERR_INVALID_BRANCH_DIRECTION An invalid branching direction is specified.
1430	MSK_RES_ERR_USER_FUNC_RET An user function reported an error.
1750	MSK_RES_ERR_NAME_MAX_LEN A name is longer than the buffer that is supposed to hold it.
1305	MSK_RES_ERR_CONE_TYPE Invalid cone type specified.
4005	MSK_RES_TRM_USER_BREAK Not in use.
1256	MSK_RES_ERR_INV_BKC 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.
4015	MSK_RES_TRM_NUM_MAX_NUM_INT_SOLUTIONS The mixed-integer optimizer terminated as the maximum number of feasible solutions was reached.
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.
904	MSK_RES_WRN_ANA_ALMOST_INT_BOUNDS This warning is issued by the problem analyzer if a constraint is bound nearly integral.
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.
807	MSK_RES_WRN_CONSTRUCT_INVALID_SOL_ITG The intial value for one or more of the integer variables is not feasible.
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.

continued on next page

continued from previous page

1070	MSK_RES_ERR_BLANK_NAME An all blank name has been specified.
1252	MSK_RES_ERR_TOO_SMALL_MAXNUMANZ The maximum number of non-zeros specified for A is smaller than the number of non-zeros in the current A.
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.
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
900	MSK_RES_WRN_ANA_LARGE_BOUNDS This warning is issued by the problem analyzer, if one or more constraint or variable bounds are very large. One should consider omitting these bounds entirely by setting them to +inf or -inf.
1071	MSK_RES_ERR_DUP_NAME The same name was used multiple times for the same problem item type.
1116	MSK_RES_ERR_MPS_MUL_CSEC Multiple `CSECTION`s are given the same name.
51	MSK_RES_WRN_LARGE_BOUND A numerically large bound value is specified.
50	MSK_RES_WRN_OPEN_PARAM_FILE The parameter file could not be opened.
1291	MSK_RES_ERR_NONCONVEX

continued on next page

continued from previous page

	The optimization problem is nonconvex.
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.
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`
	Failed to connect a callback object.
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`
	The solution item number `solitem` is invalid. Please note that `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.
1090	`MSK_RES_ERR_READ_FORMAT`
	The specified format cannot be read.
1408	`MSK_RES_ERR_QCON_SUBI_TOO_SMALL`
	Invalid value in `qcsubi`.
4006	`MSK_RES_TRM_STALL`

continued on next page

continued from previous page

	The optimizer terminated due to slow progress. The most likely reason causing slow progress is that the problem is badly formulated e.g. badly scaly scaled of near infeasible. Sometimes a few dense columns in the constraint matrix can also lead to numerical problems that causes a stall. The solution returned may or may not be of acceptable quality. Therefore, the solution status should be examined to determine the status of the solution. If the solution is near optimal, then for most practical purposes the solution will be good enough. In particular, if a linear optimization problem is solved with the interior-point optimizer with basis identification turned on, the returned solution may be of acceptable quality, even in the optimizer stalled.
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.
1164	MSK_RES_ERR_LP_WRITE_GECO_PROBLEM The problem contains general convex terms that cannot be written to an LP formatted file.
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.
1380	MSK_RES_ERR_HUGE_AIJ A numerically huge value is specified for an $a_{i,j}$ element in A. The parameter MSK_DPAR_DATA_TOL_AIJ_HUGE controls when an $a_{i,j}$ is considered huge.
71	MSK_RES_WRN_MPS_SPLIT_RAN_VECTOR

continued on next page

continued from previous page

	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.
3105	`MSK_RES_ERR_AD_MISSING_OPERAND`
	The code list data was invalid. Missing operand for operator.
1111	`MSK_RES_ERR_MPS_SPLITTED_VAR`
	All elements in a column of the A matrix must be specified consecutively. Hence, it is illegal to specify non-zero elements in A for variable 1, then for variable 2 and then variable 1 again.
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.
1159	`MSK_RES_ERR_READ_LP_MISSING_END_TAG`
	Missing End tag in LP file.
4001	`MSK_RES_TRM_MAX_TIME`
	The optimizer terminated at the maximum amount of time.
810	`MSK_RES_WRN_CONSTRUCT_NO_SOL_ITG`
	The construct solution requires an integer solution.
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.
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.
1269	`MSK_RES_ERR_INV_SK_STR`
	Invalid status key string encountered.
1036	`MSK_RES_ERR_NEWER_DLL`
	The dynamic link library is newer than the specified version.
1251	`MSK_RES_ERR_NUMVARLIM`
	Maximum number of variables limit is exceeded.
1113	`MSK_RES_ERR_MPS_MUL_QSEC`

continued on next page

continued from previous page

	Multiple QSECTIONs are specified for a constraint in the MPS data file.
502	MSK_RES_WRN_EMPTY_NAME 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.
1272	MSK_RES_ERR_INV_CONE_TYPE Invalid cone type code is encountered.
1102	MSK_RES_ERR_MPS_INV_MARKER An invalid marker has been specified in the MPS file.
1230	MSK_RES_ERR_INF_DOU_NAME A double information name is invalid.
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.
3104	MSK_RES_ERR_AD_INVALID_OPERAND The code list data was invalid. An unknown operand was used.
1015	MSK_RES_ERR_LICENSE_SERVER The license server is not responding.
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.
1450	MSK_RES_ERR_NAN_IN_DOUBLE_DATA An invalid floating point value was used in some double data.
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.

continued on next page

continued from previous page

351	MSK_RES_WRN_SOL_FILE_IGNORED_CON One or more lines in the constraint section were ignored when reading a solution file.
902	MSK_RES_WRN_ANA_EMPTY_COLS This warning is issued by the problem analyzer, if columns, in which all coefficients are zero, are found.
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.
1405	MSK_RES_ERR_INV_QCON_SUBI Invalid value in `qcsubi`.
1760	MSK_RES_ERR_NAME_IS_NULL The name buffer is a NULL pointer.
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.
1169	MSK_RES_ERR_OPF_NEW_VARIABLE Introducing new variables is now allowed. When a `[variables]` section is present, it is not allowed to introduce new variables later in the problem.
1014	MSK_RES_ERR_FLEXLM The FLEXlm license manager reported an error.
1275	MSK_RES_ERR_INVALID_SURPLUS Invalid surplus.
65	MSK_RES_WRN_NAME_MAX_LEN A name is longer than the buffer that is supposed to hold it.
1301	MSK_RES_ERR_CONE_SIZE

continued on next page

continued from previous page

	A cone with too few members is specified.
1261	`MSK_RES_ERR_FIRST`
	Invalid `first`.
1473	`MSK_RES_ERR_NAN_IN_AIJ`
	$a_{i,j}$ contains an invalid floating point value, i.e. a `NaN`.
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.
1409	`MSK_RES_ERR_QCON_SUBI_TOO_LARGE`
	Invalid value in `qcsubi`.
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.
4000	`MSK_RES_TRM_MAX_ITERATIONS`
	The optimizer terminated at the maximum number of iterations.
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.
3103	`MSK_RES_ERR_AD_INVALID_OPERATOR`
	The code list data was invalid. An unknown operator was used.
1166	`MSK_RES_ERR_WRITING_FILE`
	An error occurred while writing file
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.
3800	`MSK_RES_ERR_INT64_TO_INT32_CAST`
	An 32 bit integer could not cast to a 64 bit integer.
1552	`MSK_RES_ERR_NO_OPTIMIZER_VAR_TYPE`
	No optimizer is available for this class of optimization problems.

I.41 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 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.42 Scaling type

Value	Name Description
0	MSK_SCALING_METHOD_POW2 Scales only with power of 2 leaving the mantissa untouched.
1	MSK_SCALING_METHOD_FREE The optimizer chooses the scaling heuristic.

I.43 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.44 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.45 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 The simplex optimizer should use an aggressive degeneration strategy.
1	MSK_SIM_DEGEN_FREE The simplex optimizer chooses the degeneration strategy.

I.46 Exploit duplicate columns.

Value	Name Description
1	MSK_SIM_EXPLOIT_DUPVEC_ON Allow the simplex optimizer to exploit duplicated columns.
0	MSK_SIM_EXPLOIT_DUPVEC_OFF Disallow the simplex optimizer to exploit duplicated columns.
2	MSK_SIM_EXPLOIT_DUPVEC_FREE The simplex optimizer can choose freely.

I.47 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

continued on next page

continued from previous page

	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.48 Problem reformulation.

Value	Name Description
1	MSK_SIM_REFORMULATION_ON Allow the simplex optimizer to reformulate the problem.
3	MSK_SIM_REFORMULATION_AGGRESSIVE The simplex optimizer should use an aggressive reformulation strategy.
0	MSK_SIM_REFORMULATION_OFF Disallow the simplex optimizer to reformulate the problem.
2	MSK_SIM_REFORMULATION_FREE The simplex optimizer can choose freely.

I.49 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.50 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.51 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.52 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.53 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.54 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.55 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.56 Starting point types

Value	Name Description
1	MSK_STARTING_POINT_GUESS The optimizer guesses a starting point.
3	MSK_STARTING_POINT_SATISFY_BOUNDS The starting point is choosen to satisfy all the simple bounds on non-linear variables. If this starting point is employed, then more care than usual should employed when choosing the bounds on the non-linear variables. In particular very tight bounds should be avoided.
2	MSK_STARTING_POINT_CONSTANT The optimizer constructs a starting point by assigning a constant value to all primal and dual variables. This starting point is normally robust.
0	MSK_STARTING_POINT_FREE The starting point is chosen automatically.

I.57 Stream types

Value	Name Description
1	MSK_STREAM_MSG Message stream. Log information relating to performance and progress of the optimization is written to this stream.
3	MSK_STREAM_WRN Warning stream. Warning messages are written to this stream.
0	MSK_STREAM_LOG

continued on next page

continued from previous page

	Log stream. Contains the aggregated contents of all other streams. This means that a message written to any other stream will also be written to this stream.
2	MSK_STREAM_ERR Error stream. Error messages are written to this stream.

I.58 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.59 Variable types

Value	Name Description
1	MSK_VAR_TYPE_INT Is an integer variable.
0	MSK_VAR_TYPE_CONT Is a continuous variable.

I.60 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.

Appendix J

Problem analyzer examples

This appendix presents a few examples of the output produced by the problem analyzer described in Section 12.1. The first two problems are taken from the MIPLIB 2003 collection, http://miplib.zib.de/.

J.1 air04

```
Analyzing the problem

Constraints               Bounds                    Variables
 fixed   : all             ranged  : all             bin : all

-------------------------------------------------------------------------------

Objective, min cx
   range: min |c|: 31.0000     max |c|: 2258.00
 distrib:        |c|        vars
           [31, 100)         176
        [100, 1e+03)        8084
   [1e+03, 2.26e+03]         644

-------------------------------------------------------------------------------

Constraint matrix A has
       823 rows (constraints)
      8904 columns (variables)
     72965 (0.995703%) nonzero entries (coefficients)

Row nonzeros, A_i
   range: min A_i: 2 (0.0224618%)    max A_i: 368 (4.13297%)
 distrib:         A_i        rows       rows%        acc%
                    2           2        0.24        0.24
               [3, 7]           4        0.49        0.73
              [8, 15]          19        2.31        3.04
             [16, 31]          80        9.72       12.76
             [32, 63]         236       28.68       41.43
            [64, 127]         289       35.12       76.55
           [128, 255]         186       22.60       99.15
```

```
         [256, 368]          7        0.85      100.00

Column nonzeros, A|j
   range: min A|j: 2 (0.243013%)    max A|j: 15 (1.8226%)
 distrib:        A|j        cols       cols%        acc%
                   2         118        1.33        1.33
              [3, 7]        2853       32.04       33.37
             [8, 15]        5933       66.63      100.00

A nonzeros, A(ij)
   range: all |A(ij)| = 1.00000

-------------------------------------------------------------------------------

Constraint bounds, lb <= Ax <= ub
 distrib:        |b|            lbs             ubs
             [1, 10]            823             823

Variable bounds, lb <= x <= ub
 distrib:        |b|            lbs             ubs
                   0           8904
             [1, 10]                           8904

-------------------------------------------------------------------------------
```

J.2 arki001

```
Analyzing the problem

Constraints               Bounds                    Variables
 lower bd:        82       lower bd:        38       cont:       850
 upper bd:       946       fixed   :       353       bin :       415
 fixed   :        20       free    :         1       int :       123
                           ranged  :       996

-------------------------------------------------------------------------------

Objective, min cx
   range: all |c| in {0.00000, 1.00000}
 distrib:        |c|        vars
                   0        1387
                   1           1

-------------------------------------------------------------------------------

Constraint matrix A has
      1048 rows (constraints)
      1388 columns (variables)
     20439 (1.40511%) nonzero entries (coefficients)

Row nonzeros, A_i
   range: min A_i: 1 (0.0720461%)    max A_i: 1046 (75.3602%)
 distrib:        A_i        rows       rows%        acc%
                   1          29        2.77        2.77
                   2         476       45.42       48.19
              [3, 7]          49        4.68       52.86
             [8, 15]          56        5.34       58.21
```

```
          [16, 31]          64        6.11       64.31
          [32, 63]         373       35.59       99.90
      [1024, 1046]           1        0.10      100.00

Column nonzeros, A|j
   range: min A|j: 1 (0.0954198%)    max A|j: 29 (2.76718%)
 distrib:       A|j        cols       cols%        acc%
                  1         381       27.45       27.45
                  2          19        1.37       28.82
            [3, 7]           38        2.74       31.56
           [8, 15]          233       16.79       48.34
          [16, 29]          717       51.66      100.00

A nonzeros, A(ij)
   range: min |A(ij)|: 0.000200000     max |A(ij)|: 2.33067e+07
 distrib:      A(ij)      coeffs
    [0.0002, 0.001)          167
      [0.001, 0.01)         1049
        [0.01, 0.1)         4553
           [0.1, 1)         8840
            [1, 10)         3822
          [10, 100)          630
      [100, 1e+03)           267
    [1e+03, 1e+04)           699
    [1e+04, 1e+05)           291
    [1e+05, 1e+06)            83
    [1e+06, 1e+07)            19
 [1e+07, 2.33e+07]            19

-------------------------------------------------------------------------------

Constraint bounds, lb <= Ax <= ub
 distrib:       |b|            lbs             ubs
           [0.1, 1)                            386
            [1, 10)                             74
          [10, 100)            101             456
        [100, 1000)                             34
      [1000, 10000)                             15
    [100000, 1e+06)              1               1

Variable bounds, lb <= x <= ub
 distrib:       |b|            lbs             ubs
                  0            974             323
      [0.001, 0.01)                             19
           [0.1, 1)            370              57
            [1, 10)             41             704
          [10, 100]              2             246

-------------------------------------------------------------------------------
```

J.3 Problem with both linear and quadratic constraints

```
Analyzing the problem

Constraints              Bounds                    Variables
 lower bd:        40       upper bd:        1       cont: all
 upper bd:       121       fixed   :      204
```

```
 fixed   :      5480       free    :      5600
 ranged  :       161       ranged  :        40

-------------------------------------------------------------------------------

Objective, maximize cx
   range: all |c| in {0.00000, 15.4737}
 distrib:        |c|       vars
                   0       5844
            15.4737           1

-------------------------------------------------------------------------------

Constraint matrix A has
      5802 rows (constraints)
      5845 columns (variables)
      6480 (0.0191079%) nonzero entries (coefficients)

Row nonzeros, A_i
   range: min A_i: 0 (0%)    max A_i: 3 (0.0513259%)
 distrib:        A_i       rows       rows%        acc%
                   0         80        1.38        1.38
                   1       5003       86.23       87.61
                   2        680       11.72       99.33
                   3         39        0.67      100.00
0/80 empty rows have quadratic terms

Column nonzeros, A|j
   range: min A|j: 0 (0%)    max A|j: 15 (0.258532%)
 distrib:        A|j       cols       cols%        acc%
                   0        204        3.49        3.49
                   1       5521       94.46       97.95
                   2         40        0.68       98.63
              [3, 7]         40        0.68       99.32
             [8, 15]         40        0.68      100.00
0/204 empty columns correspond to variables used in conic
 and/or quadratic expressions only

A nonzeros, A(ij)
   range: min |A(ij)|: 2.02410e-05     max |A(ij)|: 35.8400
 distrib:      A(ij)      coeffs
  [2.02e-05, 0.0001)          40
     [0.0001, 0.001)         118
       [0.001, 0.01)         305
         [0.01, 0.1)         176
            [0.1, 1)          40
             [1, 10)        5721
          [10, 35.8]          80

-------------------------------------------------------------------------------

Constraint bounds, lb <= Ax <= ub
 distrib:        |b|           lbs            ubs
                   0          5481           5600
       [1000, 10000)                            1
     [10000, 100000)             2              1
      [1e+06, 1e+07)            78             40
      [1e+08, 1e+09]           120            120
```

```
Variable bounds, lb <= x <= ub
 distrib:        |b|           lbs            ubs
                   0           243            203
           [0.1, 1)              1              1
      [1e+06, 1e+07)                           40
      [1e+11, 1e+12]                            1

-------------------------------------------------------------------------------

Quadratic constraints: 121

Gradient nonzeros, Qx
   range: min Qx: 1 (0.0171086%)    max Qx: 2720 (46.5355%)
 distrib:          Qx        cons       cons%        acc%
                    1          40       33.06       33.06
          [64, 127]            80       66.12       99.17
        [2048, 2720]            1        0.83      100.00

-------------------------------------------------------------------------------
```

J.4 Problem with both linear and conic constraints

```
Analyzing the problem

Constraints               Bounds                   Variables
 upper bd:      3600       fixed   :      3601       cont: all
 fixed   :     21760       free    :     28802

-------------------------------------------------------------------------------

Objective, minimize cx
   range: all |c| in {0.00000, 1.00000}
 distrib:        |c|        vars
                   0       32402
                   1           1

-------------------------------------------------------------------------------

Constraint matrix A has
     25360 rows (constraints)
     32403 columns (variables)
     93339 (0.0113587%) nonzero entries (coefficients)

Row nonzeros, A_i
   range: min A_i: 1 (0.00308613%)    max A_i: 8 (0.0246891%)
 distrib:        A_i        rows       rows%        acc%
                   1        3600       14.20       14.20
                   2       10803       42.60       56.79
              [3, 7]        3995       15.75       72.55
                   8        6962       27.45      100.00

Column nonzeros, A|j
   range: min A|j: 0 (0%)    max A|j: 61 (0.240536%)
 distrib:        A|j        cols       cols%        acc%
                   0        3602       11.12       11.12
```

```
                   1       10800       33.33       44.45
                   2        7200       22.22       66.67
              [3, 7]        7279       22.46       89.13
             [8, 15]        3521       10.87      100.00
            [32, 61]           1        0.00      100.00
3600/3602 empty columns correspond to variables used in conic
 and/or quadratic constraints only

A nonzeros, A(ij)
   range: min |A(ij)|: 0.00833333     max |A(ij)|: 1.00000
 distrib:      A(ij)      coeffs
     [0.00833, 0.01)       57280
         [0.01, 0.1)          59
            [0.1, 1]       36000

-------------------------------------------------------------------------------

Constraint bounds, lb <= Ax <= ub
 distrib:        |b|            lbs             ubs
                   0          21760           21760
            [0.1, 1]                           3600

Variable bounds, lb <= x <= ub
 distrib:        |b|            lbs             ubs
             [1, 10]           3601            3601

-------------------------------------------------------------------------------


-------------------------------------------------------------------------------

Rotated quadratic cones: 3600
                 dim        RQCs
                   4        3600
```

Bibliography

[1] Richard C. Grinold abd Ronald N. Kahn. *Active portfolio management*. McGraw-Hill, New York, 2 edition, 2000.

[2] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Math. Programming*, 95(1):3–51, 2003.

[3] E. D. Andersen and K. D. Andersen. Presolving in linear programming. *Math. Programming*, 71(2):221–245, 1995.

[4] 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.

[5] 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.

[6] E. D. Andersen and Y. Ye. Combining interior-point and pivoting algorithms. *Management Sci.*, 42(12):1719–1731, December 1996.

[7] 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.

[8] E. D. Andersen and Y. Ye. On a homogeneous algorithm for the monotone complementarity problem. *Math. Programming*, 84(2):375–399, February 1999.

[9] Erling D. Andersen. The homogeneous and self-dual model and algorithm for linear optimization. Technical Report TR-1-2009, MOSEK ApS, 2009. http://www.mosek.com/fileadmin/reports/tech/homolo.pdf.

[10] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming: Theory and algorithms*. John Wiley and Sons, New York, 2 edition, 1993.

[11] A. Ben-Tal and A Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. MPS/SIAM Series on Optimization. SIAM, 2001.

[12] V. Chvátal. *Linear programming*. W.H. Freeman and Company, 1983.

[13] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL. A modeling language for mathematical programming*. Thomson, 2nd edition, 2003.

[14] N. Gould and P. L. Toint. Preprocessing for quadratic programming. *Math. Programming*, 100(1):95–132, 2004.

[15] 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.

[16] M. S. Lobo, L. Vanderberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra Appl.*, 284:193–228, November 1998.

[17] M. S. Lobo and 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.

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

[19] 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.

[20] Bernd Scherer. *Portfolio construction and risk budgeting*. Risk Books, 2 edition, 2004.

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

[22] H. P. Williams. *Model building in mathematical programming*. John Wiley and Sons, 3 edition, 1993.

[23] L. A. Wolsey. *Integer programming*. John Wiley and Sons, 1998.

Index