The MOSEK command line tool.
Version 7.1 (Revision 31).

www.mosek.com

- Published by MOSEK ApS, Denmark.

Contents

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 Ō Denmark	

License agreement

Before using the MOSEK software, please read the license agreement available in the distribution at `mosek\7\license.pdf`

Chapter 1

Changes and new features in MOSEK

The section presents improvements and new features added to MOSEK in version 7.

1.1 Platform support

In Table 1.1 the supported platform and compiler used to build MOSEK shown. Although RedHat is explicitly mentioned as the supported Linux distribution then MOSEK will work on most other variants of Linux. However, the license manager tools requires Linux Standard Base 3 or newer is installed.

1.2 General changes

- The interior-point optimizer has been extended to semi-definite optimization problems. Hence, MOSEK can optimize over the positive semi-definite cone.
- The network detection has been completely redesigned. MOSEK no longer try detect partial networks. The problem must be a pure primal network for the network optimizer to be used.
- The parameter iparam.objective_sense has been removed.
- The parameter iparam.intpnt_num_threads has been removed. Use the parameter iparam.num_threads instead.
- MOSEK now automatically exploit multiple CPUs i.e. the parameter iparam.num_threads is set to 0 be default. Note the amount memory that MOSEK uses grows with the number of threads employed.

Platform	OS version	C compiler
linux32x86	Redhat 5 or newer (LSB 3+)	Intel C 13.0 (gcc 4.3, glibc 2.3.4)
linux64x86	RedHat 5 or newer (LSB 3+)	Intel C 13.0 (gcc 4.3, glibc 2.3.4)
osx64x86	OSX 10.7 Lion or newer	Intel C 13.0 (llvm-gcc-4.2)
win32x86	Windows Vista, Server 2003 or newer	Intel C 13.0 (VS 2008)
win64x86	Windows Vista, Server 2003 or newer	Intel C 13.0 (VS 2008)

Interface	Supported versions
Java	Sun Java 1.6+
Microsoft.NET	2.1+
Python 2	2.6+
Python 3	3.1+

Table 1.1: Supported platforms

- The MBT file format has been replaced by a new task format. The new format supports semi-definite optimization.
- the HTML version of the documentation is no longer included in the downloads to save space. It is still available online.
- MOSEK is more restrictive about the allowed names on variables etc. This is in particular the case when writing LP files.
- MOSEK no longer tries to detect the cache sizes and is in general less sensitive to the hardware.
- The parameter is set iparam.auto_update_sol_info is default off. In previous version it was by default on.
- The function relaxprimal has been deprecated and replaced by the function primalrepair.

1.3 Optimizers

1.3.1 Interior point optimizer

- The factorization routines employd by the interior-point optimizer for linear and conic optimization problems has been completely rewritten. In particular the dense column detection and handling is improved. The factorization routine will also exploit vendor tuned BLAS routines.

1.3.2 The simplex optimizers

- No major changes.

1.3.3 Mixed-integer optimizer

- A new mixed-integer for linear and conic problems has been introduced. It is from run-to-run determinitic and is parallelized. It is particular suitable for conic problems.

1.4 Optimization toolbox for MATLAB

- A MOSEK equivalent of `bintprog` has been introduced.
- The functionality of the MOSEK version of `linprog` has been improved. It is now possible to employ the simplex optimizer in `linprog`.
- `mosekopt` now accepts a dense A matrix.
- An new method for specification of cones that is more efficient when the problem has many cones has introduced. The old method is still allowed but is deprecated.
- Support for semidefinite optimization problems has been added to the toolbox.

1.5 License system

- Flexlm has been upgraded to version 11.11.

1.6 Other changes

- The documentation has been improved.

1.7 Interfaces

- Semi-definite optimization capabilities have been add to the optimizer APIs.
- A major clean up have occured in the optimizer APIs. This should have little effect for most users.
- A new object orientated interface called Fusion has been added. Fusion is available Java, MATLAB, .NET and Python.
- The AMPL command line tool has been updated to the latest version.

1.8 Platform changes

- 32 bit MAC OSX on Intel x86 (osx32x86) is no longer supported.
- 32 and 64 bit Solaris on Intel x86 (solaris32x86,solaris64x86) is no longer supported.

Chapter 2

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 classes 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 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 be invoked from various programming languages.
  - C/C++, Delphi and similar languages.
  - C# (and other .NET languages),
  - Java and
  - Python

  Furthermore, the MOSEK Optimization Toolbox for MATLAB allows the MOSEK solvers to be used from Matlab.
- Third party modeling languages:
  - AMPL: 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.

    See http://www.ampl.com.
  - GAMS: Another high level modeling language for formulating optimization problems in a clean algebraic way.

Chapter 3

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 [1], http://ampl.com/BOOK/download.html 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.

3.1 Invoking the AMPL shell

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

```
mampl
```

3.2 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 4.5.

3.3 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
```

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 10 for all MOSEK response codes.

Figure 3.1: Interpretation of `solve_result_num`.

when the AMPL shell has started type the commands:

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

3.4 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 3.1 for possible values of this parameter.

3.5 Optimizer options

3.5.1 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 9 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.

3.5.2 Options

3.5.2.1 `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';
```

3.5.2.2 `wantsol`

MOSEK recognize the option `wantsol`. We refer the reader to the AMPL manual [1] for details about this option.

3.6 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 auxfiles rc;
```

before the

```
solve;
```

command makes MOSEK obtain the constraint and variable names automatically.

3.7 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.

3.8 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.

3.9 The infeasibility report

For linear optimization problems without any integer constrained variables MOSEK can generate an infeasibility report automatically. The report provides important information about the infeasibility.

The generation of the infeasibility report is turned on using the parameter setting

```
option auxfiles rc;
option mosek_options 'msk_ipar_infeas_report_auto=msk_on';
```

For further details about infeasbility report see Section 7.2.

3.10 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
;
```

3.11 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 4

Problem formulation and solutions

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.

4.1 Linear optimization

A linear optimization problem can be written as

$$\begin{array}{lccccc} \text{minimize} & & & c^T x + c^f & & \\ \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \end{array} \tag{4.1}$$

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 (4.1). If (4.1) has at least one primal feasible solution, then (4.1) is said to be (primal) feasible.

In case (4.1) does not have a feasible solution, the problem is said to be *(primal) infeasible* .

4.1.1 Duality for linear optimization

Corresponding to the primal problem (4.1), there is a dual problem

$$\begin{array}{lccl} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f & & \\ \text{subject to} & A^T y + s_l^x - s_u^x & = & c, \\ & -y + s_l^c - s_u^c & = & 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0. \end{array} \tag{4.2}$$

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 (4.2). If (4.2) has at least one feasible solution, then (4.2) is *(dual) feasible*, otherwise the problem is *(dual) infeasible*.

4.1.1.1 A primal-dual feasible solution

A solution

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

is denoted a *primal-dual feasible solution*, if (x) is a solution to the primal problem (4.1) and $(y, s_l^c, s_u^c, s_l^x, s_u^x)$ is a solution to the corresponding dual problem (4.2).

4.1.1.2 The duality gap

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 *duality gap* as the difference between the primal and the dual objective value,

$$\begin{aligned} & c^T x^* + c^f - \left((l^c)^T (s_l^c)^* - (u^c)^T (s_u^c)^* + (l^x)^T (s_l^x)^* - (u^x)^T (s_u^x)^* + c^f\right) \\ = & \sum_{i=0}^{m-1} [(s_l^c)_i^* ((x_i^c)^* - l_i^c) + (s_u^c)_i^* (u_i^c - (x_i^c)^*)] + \sum_{j=0}^{n-1} \left[(s_l^x)_j^* (x_j - l_j^x) + (s_u^x)_j^* (u_j^x - x_j^*)\right] \\ & \geq 0 \end{aligned} \tag{4.3}$$

where the first relation can be obtained by transposing and multiplying the dual constraints (4.2) 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.

4.1.1.3 When the objective is to be maximized

When the objective sense of problem (4.1) is maximization, i.e.

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

the objective sense of the dual problem changes to minimization, and the domain of all dual variables changes sign in comparison to (4.2). The dual problem thus takes the form

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

This means that the duality gap, defined in (4.3) as the primal minus the dual objective value, becomes nonpositive. It follows that the dual objective will always be greater than or equal to the primal objective.

4.1.1.4 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*

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

are satisfied.

If (4.1) 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.

4.1.2 Infeasibility for linear optimization

4.1.2.1 Primal infeasible problems

If the problem (4.1) is infeasible (has no feasible solution), MOSEK will report a certificate of primal infeasibility: The dual solution reported is the certificate of infeasibility, and the primal solution is undefined.

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

$$
\begin{array}{ll}
\text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x \\
\text{subject to} & \begin{array}{rcl} A^T y + s_l^x - s_u^x & = & 0, \\ -y + s_l^c - s_u^c & = & 0, \\ s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0, \end{array}
\end{array}
\tag{4.4}
$$

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

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

to (4.4) 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 (4.4) is unbounded, and that its dual is infeasible. As the constraints to the dual of (4.4) is identical to the constraints of problem (4.1), we thus have that problem (4.1) is also infeasible.

4.1.2.2 Dual infeasible problems

If the problem (4.2) is infeasible (has no feasible solution), MOSEK will report a certificate of dual infeasibility: The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

$$
\begin{array}{lccccc}
\text{minimize} & & & c^T x & & \\
\text{subject to} & \hat{l}^c & \leq & Ax & \leq & \hat{u}^c, \\
 & \hat{l}^x & \leq & x & \leq & \hat{u}^x,
\end{array}
\tag{4.5}
$$

where

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

and

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

such that the objective value $c^T x$ is strictly negative.

Such a solution implies that (4.5) is unbounded, and that its dual is infeasible. As the constraints to the dual of (4.5) is identical to the constraints of problem (4.2), we thus have that problem (4.2) is also infeasible.

4.1.2.3 Primal and dual infeasible case

In case that both the primal problem (4.1) and the dual problem (4.2) are infeasible, MOSEK will report only one of the two possible certificates — which one is not defined (MOSEK returns the first certificate found).

4.2 Conic quadratic optimization

Conic quadratic optimization is an extensions of linear optimization (see Section 4.1) allowing conic domains to be specified for subsets of the problem variables. A conic quadratic optimization problem can be written as

$$\begin{array}{lccccc} \text{minimize} & & & c^T x + c^f & & \\ \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \\ & & & x \in \mathcal{C}, & & \end{array} \tag{4.6}$$

where set $\mathcal{C}$ is a Cartesian product of convex cones, namely $\mathcal{C} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_p$. Having the domain restriction, $x \in \mathcal{C}$, is thus equivalent to

$$x^t \in \mathcal{C}_t \subseteq \mathbb{R}^{n_t},$$

where $x = (x^1, \ldots, x^p)$ is a partition of the problem variables. 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:

- The $\mathbb{R}^n$ set.

- The quadratic cone:

$$\mathcal{Q}_n = \left\{ x \in \mathbb{R}^n : x_1 \geq \sqrt{\sum_{j=2}^{n} x_j^2} \right\}.$$

- The rotated quadratic cone:

$$\mathcal{Q}_n^r = \left\{ x \in \mathbb{R}^n : 2x_1x_2 \geq \sum_{j=3}^{n} x_j^2,\ x_1 \geq 0,\ x_2 \geq 0 \right\}.$$

Although these cones may seem to provide only limited expressive power they can be used to model a wide range of problems as demonstrated in [2].

4.2.1 Duality for conic quadratic optimization

The dual problem corresponding to the conic quadratic optimization problem (4.6) is given by

$$\begin{array}{ll} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f \\ \text{subject to} & \begin{array}{rcl} A^T y + s_l^x - s_u^x + s_n^x & = & c, \\ -y + s_l^c - s_u^c & = & 0, \\ s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0, \\ s_n^x \in \mathcal{C}^*, & & \end{array} \end{array} \tag{4.7}$$

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

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

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:

- The $\mathbb{R}^n$ set:

$$\mathcal{C}_t = \mathbb{R}^{n_t} \ \Leftrightarrow\ \mathcal{C}_t^* = \{ s \in \mathbb{R}^{n_t} : \ s = 0 \}.$$

- The quadratic cone:

$$\mathcal{C}_t = \mathcal{Q}_{n_t} \ \Leftrightarrow\ \mathcal{C}_t^* = \mathcal{Q}_{n_t} = \left\{ s \in \mathbb{R}^{n_t} : s_1 \geq \sqrt{\sum_{j=2}^{n_t} s_j^2} \right\}.$$

- The rotated quadratic cone:

$$\mathcal{C}_t = \mathcal{Q}_{n_t}^r \ \Leftrightarrow\ \mathcal{C}_t^* = \mathcal{Q}_{n_t}^r = \left\{ s \in \mathbb{R}^{n_t} : 2s_1s_2 \geq \sum_{j=3}^{n_t} s_j^2,\ s_1 \geq 0,\ s_2 \geq 0 \right\}.$$

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

4.2.2 Infeasibility for conic quadratic optimization

In case MOSEK finds a problem to be infeasible it reports a certificate of the infeasibility. This works exactly as for linear problems (see Section 4.1.2).

4.2.2.1 Primal infeasible problems

If the problem (4.6) is infeasible, MOSEK will report a certificate of primal infeasibility: The dual solution reported is the certificate of infeasibility, and the primal solution is undefined.

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

$$\begin{array}{ll} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x \\ \text{subject to} & \begin{array}{lcl} A^T y + s_l^x - s_u^x + s_n^x & = & 0, \\ -y + s_l^c - s_u^c & = & 0, \\ s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0, \\ s_n^x \in \mathcal{C}^*, & & \end{array} \end{array} \tag{4.8}$$

such that the objective value is strictly positive.

4.2.2.2 Dual infeasible problems

If the problem (4.7) is infeasible, MOSEK will report a certificate of dual infeasibility: The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

$$\begin{array}{lccccc} \text{minimize} & & & c^T x & & \\ \text{subject to} & \hat{l}^c & \leq & Ax & \leq & \hat{u}^c, \\ & \hat{l}^x & \leq & x & \leq & \hat{u}^x, \\ & & & x \in \mathcal{C}, & & \end{array} \tag{4.9}$$

where

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

and

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

such that the objective value is strictly negative.

4.3 Semidefinite optimization

Semidefinite optimization is an extension of conic quadratic optimization (see Section 4.2) allowing positive semidefinite matrix variables to be used in addition to the usual scalar variables. A semidefinite optimization problem can be written as

$$
\begin{array}{lccccl}
\text{minimize} & & & \sum_{j=0}^{n-1} c_j x_j + \sum_{j=0}^{p-1} \left\langle \overline{C}_j, \overline{X}_j \right\rangle + c^f & & \\
\text{subject to} & l_i^c & \leq & \sum_{j=0}^{n-1} a_{ij} x_j + \sum_{j=0}^{p-1} \left\langle \overline{A}_{ij}, \overline{X}_j \right\rangle & \leq & u_i^c, \quad i = 0, \ldots, m-1 \\
 & l_j^x & \leq & x_j & \leq & u_j^x, \quad j = 0, \ldots, n-1 \\
 & & & x \in \mathcal{C}, \overline{X}_j \in \mathcal{S}_{r_j}^+, & & \quad\ \ j = 0, \ldots, p-1
\end{array}
\tag{4.10}
$$

where the problem has p symmetric positive semidefinite variables $\overline{X}_j \in \mathcal{S}_{r_j}^+$ of dimension r_j with symmetric coefficient matrices $\overline{C}_j \in \mathcal{S}_{r_j}$ and $\overline{A}_{i,j} \in \mathcal{S}_{r_j}$. We use standard notation for the matrix inner product, i.e., for $U, V \in \mathbb{R}^{m \times n}$ we have

$$\langle U, V \rangle := \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} U_{ij} V_{ij}.$$

With semidefinite optimization we can model a wide range of problems as demonstrated in [2].

4.3.1 Duality for semidefinite optimization

The dual problem corresponding to the semidefinite optimization problem (4.10) is given by

$$
\begin{array}{llcll}
\text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f & & & \\
\text{subject to} & c - A^T y + s_u^x - s_l^x & = & s_n^x, & \\
 & \overline{C}_j - \sum_{i=0}^{m} y_i \overline{A}_{ij} & = & \overline{S}_j, & j = 0, \ldots, p-1 \\
 & s_l^c - s_u^c & = & y, & \\
 & s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0, & \\
 & s_n^x \in \mathcal{C}^*, \ \overline{S}_j \in \mathcal{S}_{r_j}^+, & & & j = 0, \ldots, p-1
\end{array}
\tag{4.11}
$$

where $A \in \mathbb{R}^{m \times n}$, $A_{ij} = a_{ij}$, which is similar to the dual problem for conic quadratic optimization (see Section 4.7), except for the addition of dual constraints

$$(\overline{C}_j - \sum_{i=0}^{m} y_i \overline{A}_{ij}) \in \mathcal{S}_{r_j}^+.$$

Note that the dual of the dual problem is identical to the original primal problem.

4.3.2 Infeasibility for semidefinite optimization

In case MOSEK finds a problem to be infeasible it reports a certificate of the infeasibility. This works exactly as for linear problems (see Section 4.1.2).

4.3.2.1 Primal infeasible problems

If the problem (4.10) is infeasible, 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 problem

$$
\begin{array}{lllll}
\text{maximize} & & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x & & \\
\text{subject to} & A^T y + s_l^x - s_u^x + s_n^x & = & 0, & \\
 & \sum_{i=0}^{m-1} y_i \overline{A}_{ij} + \overline{S}_j & = & 0, & j = 0, \ldots, p-1 \\
 & -y + s_l^c - s_u^c & = & 0, & \\
 & s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0, & \\
 & s_n^x \in \mathcal{C}^*, \ \overline{S}_j \in \mathcal{S}_{r_j}^+, & & & j = 0, \ldots, p-1
\end{array}
\tag{4.12}
$$

such that the objective value is strictly positive.

4.3.2.2 Dual infeasible problems

If the problem (4.11) is infeasible, MOSEK will report a certificate of dual infeasibility: The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

$$
\begin{array}{llllll}
\text{minimize} & & & \sum_{j=0}^{n-1} c_j x_j + \sum_{j=0}^{p-1} \left\langle \overline{C}_j, \overline{X}_j \right\rangle & & \\
\text{subject to} & \hat{l}_i^c & \leq & \sum_{j=1} a_{ij} x_j + \sum_{j=0}^{p-1} \left\langle \overline{A}_{ij}, \overline{X}_j \right\rangle & \leq & \hat{u}_i^c, \quad i = 0, \ldots, m-1 \\
 & \hat{l}^x & \leq & x & \leq & \hat{u}^x, \\
 & & & x \in \mathcal{C}, \ \overline{X}_j \in \mathcal{S}_{r_j}^+, & & \quad j = 0, \ldots, p-1
\end{array}
\tag{4.13}
$$

where

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

and

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

such that the objective value is strictly negative.

4.4 Quadratic and quadratically constrained optimization

A convex quadratic and quadratically constrained optimization problem is an optimization problem of the form

$$\begin{array}{lccccl} \text{minimize} & & & \frac{1}{2}x^T Q^o x + c^T x + c^f & & \\ \text{subject to} & l_k^c & \leq & \frac{1}{2}x^T Q^k x + \sum_{j=0}^{n-1} a_{kj} x_j & \leq & u_k^c, \quad k = 0, \ldots, m-1, \\ & l_j^x & \leq & x_j & \leq & u_j^x, \quad j = 0, \ldots, n-1, \end{array} \tag{4.14}$$

where Q^o and all Q^k are symmetric matrices. Moreover for convexity, Q^o must be a positive semidefinite matrix and Q^k must satisfy

$$\begin{array}{rcl} -\infty < l_k^c & \Rightarrow & Q^k \text{ is negative semidefinite}, \\ u_k^c < \infty & \Rightarrow & Q^k \text{ is positive semidefinite}, \\ -\infty < l_k^c \leq u_k^c < \infty & \Rightarrow & Q^k = 0. \end{array}$$

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

Note that any convex quadratic and quadratically constrained 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 [2] for further details.

4.4.1 Duality for quadratic and quadratically constrained optimization

The dual problem corresponding to the quadratic and quadratically constrained optimization problem (4.14) is given by

$$\begin{array}{lrcl} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + \frac{1}{2}x^T \left(\sum_{k=0}^{m-1} y_k Q^k - Q^o\right) x + c^f & & \\ \text{subject to} & A^T y + s_l^x - s_u^x + \left(\sum_{k=0}^{m-1} y_k Q^k - Q^o\right) x & = & c, \\ & -y + s_l^c - s_u^c & = & 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0. \end{array} \tag{4.15}$$

The dual problem is related to the dual problem for linear optimization (see Section 4.2), but depend on variable x which in general can not be eliminated. In the solutions reported by MOSEK, the value of x is the same for the primal problem (4.14) and the dual problem (4.15).

4.4.2 Infeasibility for quadratic and quadratically constrained optimization

In case MOSEK finds a problem to be infeasible it reports a certificate of the infeasibility. This works exactly as for linear problems (see Section 4.1.2).

4.4.2.1 Primal infeasible problems

If the problem (4.14) with all $Q^k = 0$ is infeasible, MOSEK will report a certificate of primal infeasibility. As the constraints is the same as for a linear problem, the certificate of infeasibility is the same as for linear optimization (see Section 4.1.2.1).

4.4.2.2 Dual infeasible problems

If the problem (4.15) with all $Q^k = 0$ is infeasible, MOSEK will report a certificate of dual infeasibility: The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

$$\begin{array}{lccccc} \text{minimize} & & & c^T x & & \\ \text{subject to} & \hat{l}^c & \leq & Ax & \leq & \hat{u}^c, \\ & 0 & \leq & Q^o x & \leq & 0, \\ & \hat{l}^x & \leq & x & \leq & \hat{u}^x, \end{array} \tag{4.16}$$

where

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

and

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

such that the objective value is strictly negative.

4.5 General convex optimization

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

$$
\begin{array}{lccccc}
\text{minimize} & & & f(x) + c^T x + c^f & & \\
\text{subject to} & l^c & \leq & g(x) + Ax & \leq & u^c, \\
 & l^x & \leq & x & \leq & u^x,
\end{array}
\tag{4.17}
$$

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 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

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

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

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

4.5.1 Duality for general convex optimization

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

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

and the dual problem is given by

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

which is equivalent to

$$
\begin{array}{lccc}
\text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f & & \\
& +\, f(x) - g(x)^T y - (\nabla f(x)^T - \nabla g(x)^T y)^T x & & \\
\text{subject to} & A^T y + s_l^x - s_u^x - (\nabla f(x)^T - \nabla g(x)^T y) & = & c, \\
& -\, y + s_l^c - s_u^c & = & 0, \\
& s_l^c, s_u^c, s_l^x, s_u^x \geq 0. & &
\end{array}
\tag{4.18}
$$

In this context we use the following definition for scalar functions

$$
\nabla f(x) \quad = \quad \left[\frac{\partial f(x)}{\partial x_1}, \; ..., \; \frac{\partial f(x)}{\partial x_n} \right],
$$

and accordingly for vector functions

$$
\nabla g(x) \quad = \quad \left[\begin{array}{c} \nabla g_1(x) \\ \vdots \\ \nabla g_m(x) \end{array} \right].
$$

Chapter 5

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.

5.1 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.

5.1.1 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 (implied) 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.

5.1.1.1 Numerical issues in the presolve

During the presolve the problem is reformulated so that it hopefully solves faster. However, in rare cases the presolved problem may be harder to solve then the original problem. The presolve may also be infeasible although the orinal problem is not.

If it is suspected that presolved problem is much harder to solve than the original then it is suggested to first turn the eliminator off by setting the parameter `MSK_IPAR_PRESOLVE_ELIMINATOR_USE`. If that does not help, then trying to turn presolve off may help.

Since all computations are done in finite prescision then the presolve employs some tolerances when concluding a variable is fixed or constraint is redundant. If it happens that MOSEK incorrectly concludes a problem is primal or dual infeasible, then it is worthwhile to try to reduce the parameters `MSK_DPAR_PRESOLVE_TOL_X` and `MSK_DPAR_PRESOLVE_TOL_S`. However, if actually help reducing the parameters then this should be taken as an indication of the problem is badly formulated.

5.1.1.2 Eliminator

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

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

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.

In rare cases the eliminator may cause that the problem becomes much hard to solve.

5.1.1.3 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.

5.1.2 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.

5.1.3 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.

5.1.4 Using multiple threads

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 will automatically select the number of threads to be employed when solving the problem. However, the number of threads employed can be changed by setting the parameter `MSK_IPAR_NUM_THREADS`. This should never exceed the number of cores on the computer.

The speed-up obtained when using multiple threads 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 is not be worthwhile and may even be counter productive.

5.2 Linear optimization

5.2.1 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`.

5.2.2 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

$$\begin{array}{lccc} \text{minimize} & c^T x & & \\ \text{subject to} & Ax & = & b, \\ & x \geq 0. & & \end{array} \tag{5.1}$$

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 (5.1) 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

$$\begin{array}{rcl} Ax - b\tau & = & 0, \\ A^T y + s - c\tau & = & 0, \\ -c^T x + b^T y - \kappa & = & 0, \\ x, s, \tau, \kappa & \geq & 0, \end{array} \tag{5.2}$$

where y and s correspond to the dual variables in (5.1), and τ and κ are two additional scalar variables. Note that the homogeneous model (5.2) 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 (5.2) 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

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

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

$$\begin{aligned} Ax^* &= 0, \\ A^Ty^* + s^* &= 0, \\ -c^Tx^* + b^Ty^* &= \kappa^*, \\ x^*, s^*, \tau^*, \kappa^* &\geq 0. \end{aligned}$$

This implies that at least one of

$$-c^Tx^* > 0 \tag{5.3}$$

or

$$b^Ty^* > 0 \tag{5.4}$$

is satisfied. If (5.3) is satisfied then x^* is a certificate of dual infeasibility, whereas if (5.4) 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 [5].

5.2.2.1 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

$$\begin{aligned} \left\| A\frac{x^k}{\tau^k} - b \right\|_\infty &\leq \epsilon_p (1 + \|b\|_\infty), \\ \left\| A^T\frac{y^k}{\tau^k} + \frac{s^k}{\tau^k} - c \right\|_\infty &\leq \epsilon_d (1 + \|c\|_\infty), \text{ and} \\ \min\left(\frac{(x^k)^T s^k}{(\tau^k)^2}, \left| \frac{c^T x^k}{\tau^k} - \frac{b^T y^k}{\tau^k} \right| \right) &\leq \epsilon_g \max\left(1, \frac{\min(|c^T x^k|, |b^T y^k|)}{\tau^k}\right), \end{aligned} \tag{5.5}$$

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 (5.5) is that the optimizer is terminated if

- $\frac{x^k}{\tau^k}$ 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

$$-\epsilon_i c^T x^k > \frac{\|c\|_\infty}{\max(1, \|b\|_\infty)} \left\| A x^k \right\|_\infty$$

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 $\left\| Ax^k \right\|_\infty = 0$; then x^k is an exact certificate of dual infeasibility. Next assume that this is not the case, i.e.

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

and define

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

It is easy to verify that

$$\|A\bar{x}\|_\infty = \epsilon_i \frac{\max(1, \|b\|_\infty)}{\|c\|_\infty} \text{ and } -c^T \bar{x} > 1,$$

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

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 5.1: Parameters employed in termination criterion.

Finally, if

$$\epsilon_i b^T y^k > \frac{\|b\|_\infty}{\max(1, \|c\|_\infty)} \left\| A^T y^k + s^k \right\|_\infty$$

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 5.1 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 (5.5) reveals that quality of the solution is dependent on $\|b\|_\infty$ and $\|c\|_\infty$; 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 [5]. 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.

In some cases the interior-point method terminates having found a solution not too far from meeting the optimality condition (5.5). A solution is defined as *near optimal* if scaling ϵ_p, ϵ_d and ϵ_g by any number $\epsilon_n \in [1.0, +\infty]$ conditions (5.5) are satisfied.

A near optimal solution is therefore of lower quality but still potentially valuable. If for instance the solver stalls, i.e. it can make no more significant progress towards the optimal solution, a near optimal solution could be available and be good enough for the user.

The basis identification discussed in section 5.2.2.2 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.

5.2.2.2 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.

5.2.2.3 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
Optimizer  - Cones                  : 0
Optimizer  - Scalar variables       : 6                 conic                  : 0
Optimizer  - Semi-definite variables: 0                 scalarized             : 0
Factor     - setup time             : 0.00              dense det. time        : 0.00
Factor     - ML order time          : 0.00              GP order time          : 0.00
Factor     - nonzeros before factor : 3                 after factor           : 3
Factor     - dense dim.             : 0                 flops                  : 7.00e+001
ITE PFEAS    DFEAS    GFEAS    PRSTATUS   POBJ              DOBJ              MU       TIME
0   1.0e+000 8.6e+000 6.1e+000 1.00e+000  0.000000000e+000  -2.208000000e+003 1.0e+000 0.00
1   1.1e+000 2.5e+000 1.6e-001 0.00e+000  -7.901380925e+003 -7.394611417e+003 2.5e+000 0.00
2   1.4e-001 3.4e-001 2.1e-002 8.36e-001  -8.113031650e+003 -8.055866001e+003 3.3e-001 0.00
3   2.4e-002 5.8e-002 3.6e-003 1.27e+000  -7.777530698e+003 -7.766471080e+003 5.7e-002 0.01
4   1.3e-004 3.2e-004 2.0e-005 1.08e+000  -7.668323435e+003 -7.668207177e+003 3.2e-004 0.01
5   1.3e-008 3.2e-008 2.0e-009 1.00e+000  -7.668000027e+003 -7.668000015e+003 3.2e-008 0.01
6   1.3e-012 3.2e-012 2.0e-013 1.00e+000  -7.667999994e+003 -7.667999994e+003 3.2e-012 0.01
```

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 than 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 5.2.2 the columns of the iteration log has the following meaning:

- `ITE`: Iteration index.
- `PFEAS`: $\left\|Ax^k - b\tau^k\right\|_\infty$. The numbers in this column should converge monotonically towards to zero but may stall at low level due to rounding errors.
- `DFEAS`: $\left\|A^T y^k + s^k - c\tau^k\right\|_\infty$. The numbers in this column should converge monotonically toward to zero but may stall at low level due to rounding errors.
- `GFEAS`: $\left\| - cx^k + b^T y^k - \kappa^k\right\|_\infty$. The numbers in this column should converge monotonically toward to zero but may stall at low level due to rounding errors.
- `PRSTATUS`: This number converge to 1 if the problem has an optimal solution whereas it converge to -1 if that is not the case.

- 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.

5.2.3 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 5.2.4 for a discussion.

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

5.2.3.1 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 (4.1) and (4.2) 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.

5.2.3.2 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.

5.2.3.3 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.

5.2.4 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.

5.2.5 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.

5.3 Linear network optimization

5.3.1 Network flow problems

Linear optimization problems with network flow structure can often be solved significantly faster with a specialized version of the simplex method [7] than with the general solvers.

MOSEK includes a network simplex solver which frequently solves network problems significantly 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_OPTIMIZER to MSK_OPTIMIZER_NETWORK_PRIMAL_SIMPLEX.

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

5.4 Conic optimization

5.4.1 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 [8].

5.4.1.1 Interior-point termination criteria

The parameters controlling when the conic interior-point optimizer terminates are shown in Table 5.2.

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 5.2: Parameters employed in termination criterion.

5.5 Nonlinear convex optimization

5.5.1 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 [9], [10].

5.5.1.1 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.

5.5.1.2 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

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

is differentiable everywhere whereas the function

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

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

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 5.3: Parameters employed in termination criteria.

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

$$f(x) = \frac{1}{x} + \frac{1}{1-x}. \tag{5.6}$$

In this case the bounds

$$0 \leq x \leq 1$$

will not guarantee that MOSEK only evaluates 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, (5.6) can be reformulated as follows

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

5.5.1.3 Interior-point termination criteria

The parameters controlling when the general convex interior-point optimizer terminates are shown in Table 5.3.

5.6 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.

5.6.1 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.

5.6.2 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 core gets a share of the work. The user decides how many coress 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 cores are present. Therefore, the speed-up obtained when using multiple cores 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_NUM_THREADS` parameter sets the number of threads (and therefore the number of cores) that the interior point optimizer will use.

5.6.3 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

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 5.4: Default priorities for optimizer selection in concurrent optimization.

MSK_OPTIMIZER_CONCURRENT

the concurrent optimizer chosen.

The number of optimizers used in parallel is determined by the

MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS.

parameter. Moreover, the optimizers are selected according to a preassigned priority with optimizers having the highest priority being selected first. The default priority for each optimizer is shown in Table 5.6.3. 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.

5.6.3.1 Concurrent optimization from the command line

The command line

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

produces the following (edited) output:

```
...

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

...

Thread number 1 (free simplex) terminated first.

...

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

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

The line

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

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

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

Chapter 6

The optimizers for mixed-integer problems

A problem is a mixed-integer optimization problem when one or more of the variables are constrained to be integer valued. MOSEK contains two optimizers for mixed integer problems that is capable for solving mixed-integer

- linear,
- quadratic and quadratically constrained, and
- conic

problems.

Readers unfamiliar with integer optimization are recommended to consult some relevant literature, e.g. the book [11] by Wolsey.

6.1 Some concepts and facts related to mixed-integer optimization

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 . The value of 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. Even if the problem is only solved approximately, it is important to know how far the approximate solution is from an optimal one. In order to say something about the goodness of an approximate solution then the concept of a relaxation is important.

Name	Run-to-run deterministic	Parallelized	Strength	Cost
Mixed-integer conic	Yes	Yes	Conic	Free add-on
Mixed-integer	No	Partial	Linear	Payed add-on

Table 6.1: Mixed-integer optimizers.

The mixed-integer optimization problem

$$\begin{array}{rclccl} z^* & = & \text{minimize} & c^T x & & \\ & & \text{subject to} & Ax & = & b, \\ & & & x \geq 0 & & \\ & & & x_j \in \mathbb{Z}, & & \forall j \in \mathcal{J}, \end{array} \tag{6.1}$$

has the continuous relaxation

$$\begin{array}{rclccl} \underline{z} & = & \text{minimize} & c^T x & & \\ & & \text{subject to} & Ax & = & b, \\ & & & x \geq 0 & & \end{array} \tag{6.2}$$

The continuos relaxation is identical to the mixed-integer problem with the restriction that some variables must be integer removed.

There are two important observations about the continuous relaxation. Firstly, the continuous relaxation is usually much faster to optimize than the mixed-integer problem. Secondly if $\hat{x}$ is any feasible solution to (6.1) and

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

then

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

This is an important observation since if it is only possible to find a near optimal solution within a reasonable time frame then the quality of the solution can nevertheless be evaluated. The value $\underline{z}$ is a lower bound on the optimal objective value. This implies that the obtained solution is no further away from the optimum than $\bar{z} - \underline{z}$ in terms of the objective value.

Whenever a mixed-integer problem is solved MOSEK reports this lower bound so that the quality of the reported solution can be evaluated.

6.2 The mixed-integer optimizers

MOSEK includes two mixed-integer optimizers which are compared in Table 6.1. Both optimizers can handle problems with linear, quadratic objective and constraints and conic constraints. However, a problem must not contain both quadratic objective and constraints and conic constraints.

The mixed-integer conic optimizer is specialized for solving linear and conic optimization problems. It can also solve pure quadratic and quadratically constrained problems, these problems are automatically converted to conic problems before being solved. Whereas the mixed-integer optimizer deals with quadratic and quadratically constrained problems directly.

The mixed-integer conic optimizer is run-to-run deterministic. This means that if a problem is solved twice on the same computer with identical options then the obtained solution will be bit-for-bit identical for the two runs. However, if a time limit is set then this may not be case since the time taken to solve a problem is not deterministic. Moreover, the mixed-integer conic optimizer is parallelized i.e. it can exploit multiple cores during the optimization. Finally, the mixed-integer conic optimizer is a free add-on to the continuous optimizers. However, for some linear problems the mixed-integer optimizer may outperform the mixed-integer conic optimizer. On the other hand the mixed-integer conic optimizer is included with continuous optimizers free of charge and usually the fastest for conic problems.

None of the mixed-integer optimizers handles symmetric matrix variables i.e semi-definite optimization problems.

6.3 The mixed-integer conic optimizer

The mixed-integer conic optimizer is employed by setting the parameter `MSK_IPAR_OPTIMIZER` to `MSK_OPTIMIZER_MIXED_INT_CONIC`.

The mixed-integer conic employs 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.

6.3.1 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.

6.3.2 Heuristic

Initially, the integer optimizer tries to guess a good feasible solution using a heuristic.

6.3.3 The optimization phase

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

6.3.4 Caveats

The mixed-integer conic optimizer ignores the parameter

MSK_IPAR_MIO_CONT_SOL:
: The user should fix all the integer variables at their optimal value and reoptimize instead of relying in this option.

6.4 The mixed-integer optimizer

The mixed-integer optimizer is employed by setting the parameter MSK_IPAR_OPTIMIZER to MSK_OPTIMIZER_MIXED_INT. In the following it is briefly described how the 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.

6.4.1 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.

6.4.2 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.

6.4.3 The optimization phase

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

6.5 Termination criterion

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

A candidate solution that is feasible to the continuous relaxation 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.

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

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

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.

6.5.1 Relaxed termination

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 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

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 6.2: 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 6.3: Parameters affecting the termination of the integer optimizer.

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

is satisfied. If it is satisfied, the optimizer will report that the candidate solution is **near optimal** and then terminate. Please note that since this criteria depends on timing, the optimizer will not be run to run deterministic.

6.5.2 Important parameters

All δ tolerances can be adjusted using suitable parameters — see Table 6.2. In Table 6.3 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.

6.6 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 6.5 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 [11].

6.7 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 measure 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 guarantied to be within ϵ of the optimal solution.

The optimality gap can be retrieved 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`.

Chapter 7

The analyzers

7.1 The problem analyzer

The problem analyzer prints a detailed survey of the

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

of the model.

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 18 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.

7.1.1 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 18; if all entities in a section turn out to be of the same kind, the number will be replaced by `all` for brevity.

7.1.2 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.

7.1.3 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 18.

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

7.1.4 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).

7.1.5 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 18. Likewise, variables that appear in quadratic terms only will be reported as empty columns (0 nonzeros) in the linear constraint report.

7.1.6 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.

7.2 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.

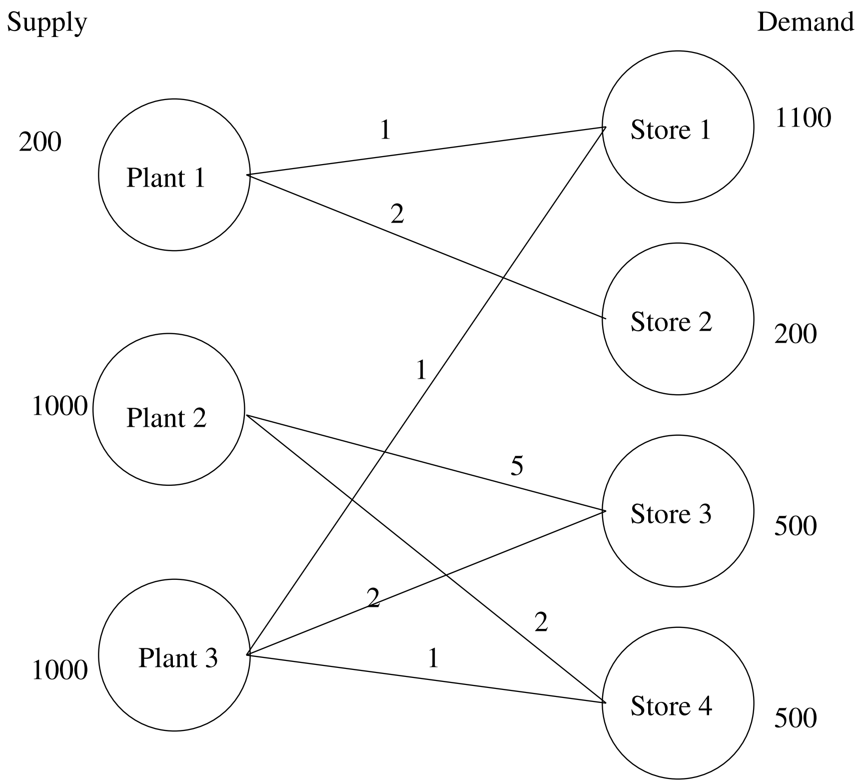

Figure 7.1: Supply, demand and cost of transportation.

7.2.1 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 7.1. The problem represented in figure 7.1 is infeasible, since the total demand

$$2300 = 1100 + 200 + 500 + 500$$

exceeds the total supply

$$2200 = 200 + 1000 + 1000$$

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:

$$
\begin{array}{lccccccccccccccccc}
\text{minimize} & x_{11} & + & 2x_{12} & + & 5x_{23} & + & 2x_{24} & + & x_{31} & + & 2x_{33} & + & x_{34} & & \\
\text{subject to} & x_{11} & + & x_{12} & & & & & & & & & & & \leq & 200, \\
 & & & & & x_{23} & + & x_{24} & & & & & & & \leq & 1000, \\
 & & & & & & & & & x_{31} & + & x_{33} & + & x_{34} & \leq & 1000, \\
 & x_{11} & & & & & & & + & x_{31} & & & & & = & 1100, \\
 & & & x_{12} & & & & & & & & & & & = & 200, \\
 & & & & & x_{23} & + & & & & & x_{33} & & & = & 500, \\
 & & & & & & & x_{24} & + & & & & & x_{34} & = & 500, \\
 & x_{ij} \geq 0. & & & & & & & & & & & & & &
\end{array}
\tag{7.1}
$$

Solving the problem (7.1) using MOSEK will result in a solution, a solution status and a problem status. Among the log output from the execution of MOSEK on the above problem are the lines:

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

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

7.2.2 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 7.2.4) 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

$$x_{12} = 200$$

makes the problem feasible.

7.2.3 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:

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

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.

7.2.3.1 A cautious note

The problem

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

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.

7.2.4 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 .

7.2.4.1 Example: Primal infeasibility

We will reuse the example (7.1) 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 (please remeber it accepts options following the C API format)

```
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 (7.1), 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 infeas.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 (7.1).

7.2.4.2 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

$$c^t x^* > 0. \tag{7.2}$$

For a minimization problem this inequality would have been reversed — see (7.5).

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 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 (7.2) 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.

7.2.5 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

$$\begin{array}{lccccc} \text{minimize} & & & c^T x + c^f & & \\ \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x \end{array} \tag{7.3}$$

where the corresponding dual problem is

$$\begin{array}{lccc} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c & & \\ & + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f & & \\ \text{subject to} & A^T y + s_l^x - s_u^x & = & c, \\ & - y + s_l^c - s_u^c & = & 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0. & & \end{array} \tag{7.4}$$

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

$$l_j^x = -\infty \;\Rightarrow\; (s_l^x)_j = 0$$

7.2.6 The certificate of primal infeasibility

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

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

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

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

and

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

The well-known Farkas Lemma tells us that (7.3) 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

$$(s_l^{c*})_i > 0((s_u^{c*})_i > 0)$$

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

$$(s_l^{x*})_j > 0((s_u^{x*})_i > 0)$$

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

7.2.7 The certificate of dual infeasibility

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

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

with negative objective value, where we use the definitions

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

and

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

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

$$\begin{array}{ccccc} & & c^T x^* & < & 0, \\ \bar{l}^c & \leq & Ax^* & \leq & \bar{u}^c, \\ \bar{l}^x & \leq & x^* & \leq & \bar{u}^x \end{array} \tag{7.5}$$

The well-known Farkas Lemma tells us that (7.4) 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

$$x_j^* \neq 0,$$

variable j is involved in the dual infeasibility.

Chapter 8

Sensitivity analysis

8.1 Introduction

Given an optimization problem it is often useful to obtain information about how the optimal objective value changes when the problem parameters are perturbed. E.g, 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 is worthwhile knowing what the value of additional capacity is. This is precisely the type of questions the sensitivity analysis deals with.

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

8.2 Restrictions

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

8.3 References

The book [12] discusses the classical sensitivity analysis in Chapter 10 whereas the book [13] presents a modern introduction to sensitivity analysis. Finally, it is recommended to read the short paper [14] to avoid some of the pitfalls associated with sensitivity analysis.

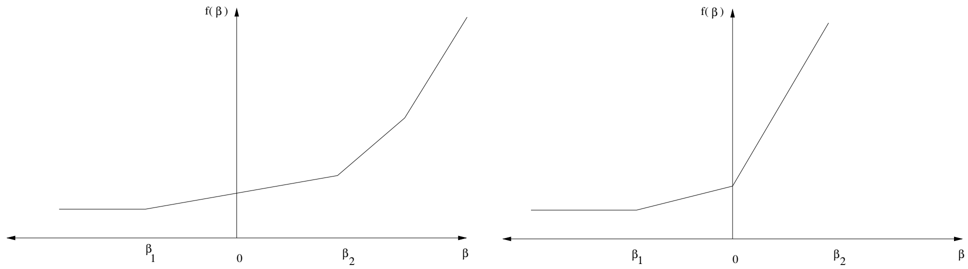

Figure 8.1: The optimal value function $f_{l_i^c}(\beta)$. Left: $\beta = 0$ is in the interior of linearity interval. Right: $\beta = 0$ is a breakpoint.

8.4 Sensitivity analysis for linear problems

8.4.1 The optimal objective value function

Assume that we are given the problem

$$\begin{array}{rcllll} z(l^c, u^c, l^x, u^x, c) & = & \text{minimize} & & c^T x & & \\ & & \text{subject to} & l^c \leq & Ax & \leq & u^c, \\ & & & & l^x \leq x \leq u^x, & & \end{array} \tag{8.1}$$

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

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

where e_i is the i th column of the identity matrix. The function

$$f_{l_i^c}(\beta) \tag{8.2}$$

shows the optimal objective value as a function of β. Please note that a change in β corresponds to a perturbation in l_i^c and hence (8.2) shows the optimal objective value as a function of l_i^c.

It is possible to prove that the function (8.2) is a piecewise linear and convex function, i.e. the function may look like the illustration in Figure 8.1. 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 the rate of change in $f_{l_i^c}(\beta)$ for small changes in β — specificly the gradient

$$f'_{l_i^c}(0),$$

which is called the *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 *linearity interval*

$$\beta \in [\beta_1, \beta_2]$$

for which

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

Since $f_{l_i^c}$ is not a smooth function $f'_{l_i^c}$ may not be defined at 0, as illustrated by the right example in figure 8.1. In this case we can define a left and a right shadow price and a left and a right linearity interval.

The function $f_{l_i^c}$ considered only changes in l_i^c. We can define similar functions for the remaining parameters of the z defined in (8.1) as well:

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

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

8.4.1.1 Equality constraints

In MOSEK a constraint can be specified as either an equality constraint or a ranged constraint. If constraint i is an equality constraint, we define the optimal value function for this as

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

Thus for an equality constraint the upper and the lower bounds (which are equal) are perturbed simultaneously. Therefore, MOSEK will handle sensitivity analysis differently for a ranged constraint with $l_i^c = u_i^c$ and for an equality constraint.

8.4.2 The basis type sensitivity analysis

The classical sensitivity analysis discussed in most textbooks about linear optimization, e.g. [12], is based on an optimal basic solution or, equivalently, on an optimal basis. This method may produce misleading results [13] 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, the basis type sensitivity analysis computes the linearity interval $[\beta_1, \beta_2]$ so 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.

8.4.3 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 of the optimal partition type sensitivity analysis is that it is computationally expensive compared to the basis type analysts. This type of sensitivity analysis is currently provided as an experimental feature in MOSEK.

Given the optimal primal and dual solutions to (8.1), i.e. x^* and $((s_l^c)^*, (s_u^c)^*, (s_l^x)^*, (s_u^x)^*)$ the optimal objective value is given by

$$z^* := c^T x^*.$$

The left and right shadow prices σ_1 and σ_2 for l_i^c are given by this pair of optimization problems:

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

and

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

These two optimization problems make it easy to interpret the shadow price. Indeed, if $((s_l^c)^*, (s_u^c)^*, (s_l^x)^*, (s_u^x)^*)$ is an arbitrary optimal solution then

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

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

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

and

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

The linearity intervals and shadow prices for u_i^c, l_j^x, and u_j^x are computed similarly to l_i^c.

The left and right shadow prices for c_j denoted σ_1 and σ_2 respectively are computed as follows:

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

and

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

Once again the above two optimization problems make it easy to interpret the shadow prices. Indeed, if x^* is an arbitrary primal optimal solution, then

$$x_j^* \in [\sigma_1, \sigma_2].$$

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

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

and

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

8.5 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
```

```
* 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 8.2: The sensitivity analysis file format.

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.

8.5.1 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 8.2, 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.
- `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

```
* 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 8.3: Example of the sensitivity file format.

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 8.4.1.1 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.

8.5.2 Example: Sensitivity analysis from command line

As an example consider the `sensitivity.ssp` file shown in Figure 8.3. The command

```
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
```

8.5.3 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.

Chapter 9

Parameters

Parameters grouped by functionality.

Analysis parameters.

Parameters controling the behaviour of the problem and solution analyzers.

- 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.

Basis identification parameters.

- MSK_IPAR_BI_CLEAN_OPTIMIZER. Controls which simplex optimizer is used in the clean-up phase.
- 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.
- MSK_IPAR_BI_MAX_ITERATIONS. Maximum number of iterations after basis identification.
- MSK_IPAR_INTPNT_BASIS. Controls whether basis identification is performed.
- MSK_IPAR_LOG_BI. Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.
- MSK_IPAR_LOG_BI_FREQ. Controls the logging frequency.
- MSK_DPAR_SIM_LU_TOL_REL_PIV. Relative pivot tolerance employed when computing the LU factorization of the basis matrix.

Behavior of the optimization task.

Parameters defining the behavior of an optimization task when loading data.

- MSK_SPAR_FEASREPAIR_NAME_PREFIX. Feasibility repair name prefix.

- MSK_SPAR_FEASREPAIR_NAME_SEPARATOR. Feasibility repair name separator.
- MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL. Feasibility repair name violation name.

Conic interior-point method parameters.

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

- MSK_DPAR_INTPNT_CO_TOL_DFEAS. Dual feasibility tolerance used by the conic interior-point optimizer.
- MSK_DPAR_INTPNT_CO_TOL_INFEAS. Infeasibility tolerance for the conic solver.
- MSK_DPAR_INTPNT_CO_TOL_MU_RED. Optimality tolerance for the conic solver.
- MSK_DPAR_INTPNT_CO_TOL_NEAR_REL. Optimality tolerance for the conic solver.
- MSK_DPAR_INTPNT_CO_TOL_PFEAS. Primal feasibility tolerance used by the conic interior-point optimizer.
- MSK_DPAR_INTPNT_CO_TOL_REL_GAP. Relative gap termination tolerance used by the conic interior-point optimizer.

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_DPAR_DATA_TOL_AIJ. Data tolerance threshold.
- MSK_DPAR_DATA_TOL_AIJ_HUGE. Data tolerance threshold.
- MSK_DPAR_DATA_TOL_AIJ_LARGE. Data tolerance threshold.
- MSK_DPAR_DATA_TOL_BOUND_INF. Data tolerance threshold.
- MSK_DPAR_DATA_TOL_BOUND_WRN. Data tolerance threshold.
- MSK_DPAR_DATA_TOL_C_HUGE. Data tolerance threshold.
- MSK_DPAR_DATA_TOL_CJ_LARGE. Data tolerance threshold.
- MSK_DPAR_DATA_TOL_QIJ. Data tolerance threshold.
- MSK_DPAR_DATA_TOL_X. Data tolerance threshold.
- MSK_IPAR_LOG_CHECK_CONVEXITY. 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.

Data input/output parameters.

Parameters defining the behavior of data readers and writers.

- MSK_SPAR_BAS_SOL_FILE_NAME. Name of the bas solution file.
- MSK_SPAR_DATA_FILE_NAME. Data are read and written to this file.
- MSK_SPAR_DEBUG_FILE_NAME. MOSEK debug file.
- MSK_SPAR_INT_SOL_FILE_NAME. Name of the int solution file.

- MSK_SPAR_ITR_SOL_FILE_NAME. Name of the itr solution file.
- MSK_IPAR_LOG_FILE. If turned on, then some log info is printed when a file is written or read.
- MSK_SPAR_MIO_DEBUG_STRING. For internal use only.
- MSK_SPAR_PARAM_COMMENT_SIGN. Solution file comment character.
- MSK_SPAR_PARAM_READ_FILE_NAME. Modifications to the parameter database is read from this file.
- MSK_SPAR_PARAM_WRITE_FILE_NAME. The parameter database is written to this file.
- MSK_SPAR_READ_MPS_BOU_NAME. Name of the BOUNDS vector used. An empty name means that the first BOUNDS vector is used.
- MSK_SPAR_READ_MPS_OBJ_NAME. Objective name in the MPS file.
- MSK_SPAR_READ_MPS_RAN_NAME. Name of the RANGE vector used. An empty name means that the first RANGE vector is used.
- MSK_SPAR_READ_MPS_RHS_NAME. Name of the RHS used. An empty name means that the first RHS vector is used.
- MSK_SPAR_SOL_FILTER_XC_LOW. Solution file filter.
- MSK_SPAR_SOL_FILTER_XC_UPR. Solution file filter.
- MSK_SPAR_SOL_FILTER_XX_LOW. Solution file filter.
- MSK_SPAR_SOL_FILTER_XX_UPR. Solution file filter.
- MSK_SPAR_STAT_FILE_NAME. Statistics file name.
- MSK_SPAR_STAT_KEY. Key used when writing the summary file.
- MSK_SPAR_STAT_NAME. Name used when writing the statistics file.
- MSK_SPAR_WRITE_LP_GEN_VAR_NAME. Added variable names in the LP files.

Debugging parameters.

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

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

Dual simplex optimizer parameters.

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

- MSK_IPAR_SIM_DUAL_CRASH. Controls whether crashing is performed in the dual simplex optimizer.
- MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION. Controls how aggressively restricted selection is used.
- MSK_IPAR_SIM_DUAL_SELECTION. Controls the dual simplex strategy.

Feasibility repair parameters.

- MSK_DPAR_FEASREPAIR_TOL. Tolerance for constraint enforcing upper bound on sum of weighted violations in feasibility repair.

Infeasibility report parameters.

- MSK_IPAR_LOG_INFEAS_ANA. Controls log level for the infeasibility analyzer.

Interior-point method parameters.

Parameters defining the behavior of the interior-point method for linear, conic and convex problems.

- MSK_IPAR_BI_IGNORE_MAX_ITER. 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.
- MSK_DPAR_CHECK_CONVEXITY_REL_TOL. Convexity check tolerance.
- MSK_IPAR_INTPNT_BASIS. Controls whether basis identification is performed.
- MSK_DPAR_INTPNT_CO_TOL_DFEAS. Dual feasibility tolerance used by the conic interior-point optimizer.
- MSK_DPAR_INTPNT_CO_TOL_INFEAS. Infeasibility tolerance for the conic solver.
- MSK_DPAR_INTPNT_CO_TOL_MU_RED. Optimality tolerance for the conic solver.
- MSK_DPAR_INTPNT_CO_TOL_NEAR_REL. Optimality tolerance for the conic solver.
- MSK_DPAR_INTPNT_CO_TOL_PFEAS. Primal feasibility tolerance used by the conic interior-point optimizer.
- MSK_DPAR_INTPNT_CO_TOL_REL_GAP. Relative gap termination tolerance used by the conic interior-point optimizer.
- MSK_IPAR_INTPNT_DIFF_STEP. Controls whether different step sizes are allowed in the primal and dual space.
- MSK_IPAR_INTPNT_MAX_ITERATIONS. Controls the maximum number of iterations allowed in the interior-point optimizer.
- MSK_IPAR_INTPNT_MAX_NUM_COR. Maximum number of correction steps.
- MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS. Maximum number of steps to be used by the iterative search direction refinement.
- MSK_DPAR_INTPNT_NL_MERIT_BAL. Controls if the complementarity and infeasibility is converging to zero at about equal rates.
- MSK_DPAR_INTPNT_NL_TOL_DFEAS. Dual feasibility tolerance used when a nonlinear model is solved.
- MSK_DPAR_INTPNT_NL_TOL_MU_RED. Relative complementarity gap tolerance.
- MSK_DPAR_INTPNT_NL_TOL_NEAR_REL. Nonlinear solver optimality tolerance parameter.
- MSK_DPAR_INTPNT_NL_TOL_PFEAS. Primal feasibility tolerance used when a nonlinear model is solved.

- MSK_DPAR_INTPNT_NL_TOL_REL_GAP. Relative gap termination tolerance for nonlinear problems.
- MSK_DPAR_INTPNT_NL_TOL_REL_STEP. Relative step size to the boundary for general nonlinear optimization problems.
- MSK_IPAR_INTPNT_OFF_COL_TRH. Controls the aggressiveness of the offending column detection.
- MSK_IPAR_INTPNT_ORDER_METHOD. Controls the ordering strategy.
- MSK_IPAR_INTPNT_REGULARIZATION_USE. Controls whether regularization is allowed.
- MSK_IPAR_INTPNT_SCALING. Controls how the problem is scaled before the interior-point optimizer is used.
- MSK_IPAR_INTPNT_SOLVE_FORM. Controls whether the primal or the dual problem is solved.
- MSK_IPAR_INTPNT_STARTING_POINT. Starting point used by the interior-point optimizer.
- MSK_DPAR_INTPNT_TOL_DFEAS. Dual feasibility tolerance used for linear and quadratic optimization problems.
- MSK_DPAR_INTPNT_TOL_DSAFE. Controls the interior-point dual starting point.
- MSK_DPAR_INTPNT_TOL_INFEAS. Nonlinear solver infeasibility tolerance parameter.
- MSK_DPAR_INTPNT_TOL_MU_RED. Relative complementarity gap tolerance.
- MSK_DPAR_INTPNT_TOL_PATH. interior-point centering aggressiveness.
- MSK_DPAR_INTPNT_TOL_PFEAS. Primal feasibility tolerance used for linear and quadratic optimization problems.
- MSK_DPAR_INTPNT_TOL_PSAFE. Controls the interior-point primal starting point.
- MSK_DPAR_INTPNT_TOL_REL_GAP. Relative gap termination tolerance.
- MSK_DPAR_INTPNT_TOL_REL_STEP. Relative step size to the boundary for linear and quadratic optimization problems.
- 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. In other words the interior-point optimizer does not make any progress and therefore it is better stop.
- MSK_IPAR_LOG_INTPNT. Controls the amount of log information from the interior-point optimizers.
- MSK_IPAR_LOG_PRESOLVE. Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.
- MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL. This parameter determines when columns are dropped in incomplete cholesky factorization doing reformulation of quadratic problems.

License manager parameters.

- MSK_IPAR_CACHE_LICENSE. Control license caching.
- MSK_IPAR_LICENSE_DEBUG. Controls the license manager client debugging behavior.
- MSK_IPAR_LICENSE_PAUSE_TIME. Controls license manager client behavior.
- MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS. Controls license manager client behavior.

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

Logging parameters.

- MSK_IPAR_LOG. Controls the amount of log information.
- MSK_IPAR_LOG_BI. Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.
- MSK_IPAR_LOG_BI_FREQ. Controls the logging frequency.
- MSK_IPAR_LOG_CONCURRENT. Controls amount of output printed by the concurrent optimizer.
- MSK_IPAR_LOG_EXPAND. Controls the amount of logging when a data item such as the maximum number constrains is expanded.
- MSK_IPAR_LOG_FACTOR. If turned on, then the factor log lines are added to the log.
- MSK_IPAR_LOG_FEAS_REPAIR. Controls the amount of output printed when performing feasibility repair. A value higher than one means extensive logging.
- MSK_IPAR_LOG_FILE. If turned on, then some log info is printed when a file is written or read.
- MSK_IPAR_LOG_HEAD. If turned on, then a header line is added to the log.
- MSK_IPAR_LOG_INFEAS_ANA. Controls log level for the infeasibility analyzer.
- 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_MIO_FREQ. The mixed-integer solver logging frequency.
- MSK_IPAR_LOG_NONCONVEX. Controls amount of output printed by the nonconvex optimizer.
- MSK_IPAR_LOG_OPTIMIZER. Controls the amount of general optimizer information that is logged.
- MSK_IPAR_LOG_ORDER. If turned on, then factor lines are added to the log.
- MSK_IPAR_LOG_PARAM. Controls the amount of information printed out about parameter changes.
- MSK_IPAR_LOG_PRESOLVE. Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.
- MSK_IPAR_LOG_RESPONSE. Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.
- MSK_IPAR_LOG_SIM. Controls the amount of log information from the simplex optimizers.
- MSK_IPAR_LOG_SIM_FREQ. Controls simplex logging frequency.
- MSK_IPAR_LOG_SIM_NETWORK_FREQ. Controls the network simplex logging frequency.
- MSK_IPAR_LOG_STORAGE. Controls the memory related log information.

Mixed-integer optimization parameters.

- MSK_IPAR_LOG_MIO. Controls the amount of log information from the mixed-integer optimizers.
- MSK_IPAR_LOG_MIO_FREQ. The mixed-integer solver logging frequency.
- MSK_IPAR_MIO_BRANCH_DIR. Controls whether the mixed-integer optimizer is branching up or down by default.
- MSK_IPAR_MIO_CONSTRUCT_SOL. Controls if an initial mixed integer solution should be constructed from the values of the integer variables.
- MSK_IPAR_MIO_CONT_SOL. Controls the meaning of interior-point and basic solutions in mixed integer problems.
- MSK_IPAR_MIO_CUT_CG. Controls whether CG cuts should be generated.
- MSK_IPAR_MIO_CUT_CMIR. Controls whether mixed integer rounding cuts should be generated.
- MSK_IPAR_MIO_CUT_LEVEL_ROOT. Controls the cut level employed by the mixed-integer optimizer at the root node.
- MSK_IPAR_MIO_CUT_LEVEL_TREE. Controls the cut level employed by the mixed-integer optimizer in the tree.
- MSK_DPAR_MIO_DISABLE_TERM_TIME. Certain termination criteria is disabled within the mixed-integer optimizer for period time specified by the parameter.
- MSK_IPAR_MIO_FEASPUMP_LEVEL. Controls the feasibility pump heuristic which is used to construct a good initial feasible solution.
- MSK_IPAR_MIO_HEURISTIC_LEVEL. Controls the heuristic employed by the mixed-integer optimizer to locate an initial integer feasible solution.
- MSK_DPAR_MIO_HEURISTIC_TIME. Time limit for the mixed-integer heuristics.
- MSK_IPAR_MIO_HOTSTART. Controls whether the integer optimizer is hot-started.
- MSK_IPAR_MIO_KEEP_BASIS. Controls whether the integer presolve keeps bases in memory.
- MSK_IPAR_MIO_MAX_NUM_BRANCHES. Maximum number of branches allowed during the branch and bound search.
- MSK_IPAR_MIO_MAX_NUM_RELAXS. Maximum number of relaxations in branch and bound search.
- MSK_IPAR_MIO_MAX_NUM_SOLUTIONS. Controls how many feasible solutions the mixed-integer optimizer investigates.
- MSK_DPAR_MIO_MAX_TIME. Time limit for the mixed-integer optimizer.
- MSK_DPAR_MIO_MAX_TIME_APRX_OPT. Time limit for the mixed-integer optimizer.
- 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.
- MSK_IPAR_MIO_NODE_OPTIMIZER. Controls which optimizer is employed at the non-root nodes in the mixed-integer optimizer.
- MSK_IPAR_MIO_NODE_SELECTION. Controls the node selection strategy employed by the mixed-integer optimizer.

- MSK_IPAR_MIO_OPTIMIZER_MODE. An exprimental feature.
- MSK_IPAR_MIO_PRESOLVE_AGGREGATE. Controls whether problem aggregation is performed in the mixed-integer presolve.
- MSK_IPAR_MIO_PRESOLVE_PROBING. Controls whether probing is employed by the mixed-integer presolve.
- MSK_IPAR_MIO_PRESOLVE_USE. Controls whether presolve is performed by the mixed-integer optimizer.
- MSK_IPAR_MIO_PROBING_LEVEL. Controls the amount of probing employed by the mixed-integer optimizer in presolve.
- MSK_DPAR_MIO_REL_ADD_CUT_LIMITED. Controls cut generation for mixed-integer optimizer.
- MSK_DPAR_MIO_REL_GAP_CONST. This value is used to compute the relative gap for the solution to an integer optimization problem.
- MSK_IPAR_MIO_RINS_MAX_NODES. Maximum number of nodes in each call to the RINS heuristic.
- MSK_IPAR_MIO_ROOT_OPTIMIZER. Controls which optimizer is employed at the root node in the mixed-integer optimizer.
- MSK_IPAR_MIO_STRONG_BRANCH. The depth from the root in which strong branching is employed.
- MSK_DPAR_MIO_TOL_ABS_GAP. Absolute optimality tolerance employed by the mixed-integer optimizer.
- MSK_DPAR_MIO_TOL_ABS_RELAX_INT. Integer constraint tolerance.
- MSK_DPAR_MIO_TOL_FEAS. Feasibility tolerance for mixed integer solver. Any solution with maximum infeasibility below this value will be considered feasible.
- MSK_DPAR_MIO_TOL_MAX_CUT_FRAC_RHS. Controls cut generation for mixed-integer optimizer.
- MSK_DPAR_MIO_TOL_MIN_CUT_FRAC_RHS. Controls cut generation for mixed-integer optimizer.
- MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT. Controls cut generation for mixed-integer optimizer.
- MSK_DPAR_MIO_TOL_REL_GAP. Relative optimality tolerance employed by the mixed-integer optimizer.
- MSK_DPAR_MIO_TOL_REL_RELAX_INT. Integer constraint tolerance.
- MSK_DPAR_MIO_TOL_X. Absolute solution tolerance used in mixed-integer optimizer.
- MSK_IPAR_MIO_USE_MULTITHREADED_OPTIMIZER. Controls wheter the new multithreaded optimizer should be used for Mixed integer problems.

Network simplex optimizer parameters.

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

- MSK_IPAR_LOG_SIM_NETWORK_FREQ. Controls the network simplex logging frequency.
- MSK_IPAR_SIM_REFACTOR_FREQ. Controls the basis refactoring frequency.

Non-convex solver parameters.

- MSK_IPAR_LOG_NONCONVEX. Controls amount of output printed by the nonconvex optimizer.
- MSK_IPAR_NONCONVEX_MAX_ITERATIONS. Maximum number of iterations that can be used by the nonconvex optimizer.
- MSK_DPAR_NONCONVEX_TOL_FEAS. Feasibility tolerance used by the nonconvex optimizer.
- MSK_DPAR_NONCONVEX_TOL_OPT. Optimality tolerance used by the nonconvex optimizer.

Nonlinear convex method parameters.

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

- MSK_DPAR_INTPNT_NL_MERIT_BAL. Controls if the complementarity and infeasibility is converging to zero at about equal rates.
- MSK_DPAR_INTPNT_NL_TOL_DFEAS. Dual feasibility tolerance used when a nonlinear model is solved.
- MSK_DPAR_INTPNT_NL_TOL_MU_RED. Relative complementarity gap tolerance.
- MSK_DPAR_INTPNT_NL_TOL_NEAR_REL. Nonlinear solver optimality tolerance parameter.
- MSK_DPAR_INTPNT_NL_TOL_PFEAS. Primal feasibility tolerance used when a nonlinear model is solved.
- MSK_DPAR_INTPNT_NL_TOL_REL_GAP. Relative gap termination tolerance for nonlinear problems.
- MSK_DPAR_INTPNT_NL_TOL_REL_STEP. Relative step size to the boundary for general nonlinear optimization problems.
- MSK_DPAR_INTPNT_TOL_INFEAS. Nonlinear solver infeasibility tolerance parameter.
- MSK_IPAR_LOG_CHECK_CONVEXITY. 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.

Optimization system parameters.

Parameters defining the overall solver system environment. This includes system and platform related information and behavior.

- MSK_IPAR_LICENSE_WAIT. Controls if MOSEK should queue for a license if none is available.
- MSK_IPAR_LOG_STORAGE. Controls the memory related log information.
- MSK_IPAR_NUM_THREADS. Controls the number of threads employed by the optimizer. If set to 0 the number of threads used will be equal to the number of cores detected on the machine.

Output information parameters.

- MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS. Controls license manager client behavior.
- MSK_IPAR_LOG. Controls the amount of log information.
- MSK_IPAR_LOG_BI. Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.

- MSK_IPAR_LOG_BI_FREQ. Controls the logging frequency.
- MSK_IPAR_LOG_EXPAND. Controls the amount of logging when a data item such as the maximum number constrains is expanded.
- MSK_IPAR_LOG_FACTOR. If turned on, then the factor log lines are added to the log.
- MSK_IPAR_LOG_FEAS_REPAIR. Controls the amount of output printed when performing feasibility repair. A value higher than one means extensive logging.
- MSK_IPAR_LOG_FILE. If turned on, then some log info is printed when a file is written or read.
- MSK_IPAR_LOG_HEAD. If turned on, then a header line is added to the log.
- MSK_IPAR_LOG_INFEAS_ANA. Controls log level for the infeasibility analyzer.
- 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_MIO_FREQ. The mixed-integer solver logging frequency.
- MSK_IPAR_LOG_NONCONVEX. Controls amount of output printed by the nonconvex optimizer.
- MSK_IPAR_LOG_OPTIMIZER. Controls the amount of general optimizer information that is logged.
- MSK_IPAR_LOG_ORDER. If turned on, then factor lines are added to the log.
- MSK_IPAR_LOG_PARAM. Controls the amount of information printed out about parameter changes.
- MSK_IPAR_LOG_RESPONSE. Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.
- MSK_IPAR_LOG_SIM. Controls the amount of log information from the simplex optimizers.
- MSK_IPAR_LOG_SIM_FREQ. Controls simplex logging frequency.
- MSK_IPAR_LOG_SIM_MINOR. Currently not in use.
- MSK_IPAR_LOG_SIM_NETWORK_FREQ. Controls the network simplex logging frequency.
- MSK_IPAR_LOG_STORAGE. Controls the memory related log information.
- MSK_IPAR_MAX_NUM_WARNINGS. A negtive number means all warnings are logged. Otherwise the parameter specifies the maximum number times each warning is logged.
- MSK_IPAR_WARNING_LEVEL. Deprecated and not in use

Overall solver parameters.

- MSK_IPAR_BI_CLEAN_OPTIMIZER. Controls which simplex optimizer is used in the clean-up phase.
- MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS. The maximum number of simultaneous optimizations that will be started by the concurrent optimizer.
- MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX. Priority of the dual simplex algorithm when selecting solvers for concurrent optimization.

- MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX. Priority of the free simplex optimizer when selecting solvers for concurrent optimization.
- MSK_IPAR_CONCURRENT_PRIORITY_INTPNT. Priority of the interior-point algorithm when selecting solvers for concurrent optimization.
- MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX. Priority of the primal simplex algorithm when selecting solvers for concurrent optimization.
- MSK_IPAR_INFEAS_PREFER_PRIMAL. Controls which certificate is used if both primal- and dual- certificate of infeasibility is available.
- MSK_IPAR_LICENSE_WAIT. Controls if MOSEK should queue for a license if none is available.
- MSK_IPAR_MIO_CONT_SOL. Controls the meaning of interior-point and basic solutions in mixed integer problems.
- MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER. Controls the size of the local search space when doing local branching.
- MSK_IPAR_MIO_MODE. Turns on/off the mixed-integer mode.
- MSK_IPAR_OPTIMIZER. Controls which optimizer is used to optimize the task.
- MSK_IPAR_PRESOLVE_LEVEL. Currently not used.
- MSK_IPAR_PRESOLVE_USE. Controls whether the presolve is applied to a problem before it is optimized.
- MSK_IPAR_SOLUTION_CALLBACK. Indicates whether solution call-backs will be performed during the optimization.

Presolve parameters.

- MSK_IPAR_PRESOLVE_ELIM_FILL. Maximum amount of fill-in in the elimination phase.
- MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES. Control the maximum number of times the eliminator is tried.
- MSK_IPAR_PRESOLVE_ELIMINATOR_USE. Controls whether free or implied free variables are eliminated from the problem.
- MSK_IPAR_PRESOLVE_LEVEL. Currently not used.
- MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH. Controls linear dependency check in presolve.
- MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH. Controls linear dependency check in presolve.
- MSK_IPAR_PRESOLVE_LINDEP_USE. Controls whether the linear constraints are checked for linear dependencies.
- MSK_DPAR_PRESOLVE_TOL_ABS_LINDEP. Absolute tolerance employed by the linear dependency checker.
- MSK_DPAR_PRESOLVE_TOL_AIJ. Absolute zero tolerance employed for constraint coefficients in the presolve.
- MSK_DPAR_PRESOLVE_TOL_REL_LINDEP. Relative tolerance employed by the linear dependency checker.
- MSK_DPAR_PRESOLVE_TOL_S. Absolute zero tolerance employed for slack variables in the presolve.

- MSK_DPAR_PRESOLVE_TOL_X. Absolute zero tolerance employed for variables in the presolve.
- MSK_IPAR_PRESOLVE_USE. Controls whether the presolve is applied to a problem before it is optimized.

Primal simplex optimizer parameters.

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

- MSK_IPAR_SIM_PRIMAL_CRASH. Controls the simplex crash.
- MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION. Controls how aggressively restricted selection is used.
- MSK_IPAR_SIM_PRIMAL_SELECTION. Controls the primal simplex strategy.

Progress call-back parameters.

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

Simplex optimizer parameters.

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

- MSK_DPAR_BASIS_REL_TOL_S. Maximum relative dual bound violation allowed in an optimal basic solution.
- MSK_DPAR_BASIS_TOL_S. Maximum absolute dual bound violation in an optimal basic solution.
- MSK_DPAR_BASIS_TOL_X. Maximum absolute primal bound violation allowed in an optimal basic solution.
- MSK_IPAR_LOG_SIM. Controls the amount of log information from the simplex optimizers.
- MSK_IPAR_LOG_SIM_FREQ. Controls simplex logging frequency.
- MSK_IPAR_LOG_SIM_MINOR. Currently not in use.
- MSK_IPAR_SIM_BASIS_FACTOR_USE. 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. Controls how aggressively degeneration is handled.
- MSK_IPAR_SIM_DUAL_PHASEONE_METHOD. An exprimental feature.
- MSK_IPAR_SIM_EXPLOIT_DUPVEC. Controls if the simplex optimizers are allowed to exploit duplicated columns.
- MSK_IPAR_SIM_HOTSTART. Controls the type of hot-start that the simplex optimizer perform.
- MSK_IPAR_SIM_INTEGER. An exprimental feature.
- MSK_DPAR_SIM_LU_TOL_REL_PIV. Relative pivot tolerance employed when computing the LU factorization of the basis matrix.
- MSK_IPAR_SIM_MAX_ITERATIONS. Maximum number of iterations that can be used by a simplex optimizer.

- MSK_IPAR_SIM_MAX_NUM_SETBACKS. Controls how many set-backs that are allowed within a simplex optimizer.
- MSK_IPAR_SIM_NON_SINGULAR. Controls if the simplex optimizer ensures a non-singular basis, if possible.
- MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD. An exprimental feature.
- MSK_IPAR_SIM_REFORMULATION. Controls if the simplex optimizers are allowed to reformulate the problem.
- MSK_IPAR_SIM_SAVE_LU. Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.
- MSK_IPAR_SIM_SCALING. Controls how much effort is used in scaling the problem before a simplex optimizer is used.
- MSK_IPAR_SIM_SCALING_METHOD. Controls how the problem is scaled before a simplex optimizer is used.
- MSK_IPAR_SIM_SOLVE_FORM. Controls whether the primal or the dual problem is solved by the primal-/dual- simplex optimizer.
- MSK_IPAR_SIM_STABILITY_PRIORITY. Controls how high priority the numerical stability should be given.
- MSK_IPAR_SIM_SWITCH_OPTIMIZER. Controls the simplex behavior.
- MSK_DPAR_SIMPLEX_ABS_TOL_PIV. Absolute pivot tolerance employed by the simplex optimizers.

Solution input/output parameters.

Parameters defining the behavior of solution reader and writer.

- MSK_SPAR_BAS_SOL_FILE_NAME. Name of the bas solution file.
- MSK_SPAR_INT_SOL_FILE_NAME. Name of the int solution file.
- MSK_SPAR_ITR_SOL_FILE_NAME. Name of the itr solution file.
- MSK_IPAR_SOL_FILTER_KEEP_BASIC. Controls the license manager client behavior.
- MSK_SPAR_SOL_FILTER_XC_LOW. Solution file filter.
- MSK_SPAR_SOL_FILTER_XC_UPR. Solution file filter.
- MSK_SPAR_SOL_FILTER_XX_LOW. Solution file filter.
- MSK_SPAR_SOL_FILTER_XX_UPR. Solution file filter.

Termination criterion parameters.

Parameters which define termination and optimality criteria and related information.

- MSK_DPAR_BASIS_REL_TOL_S. Maximum relative dual bound violation allowed in an optimal basic solution.
- MSK_DPAR_BASIS_TOL_S. Maximum absolute dual bound violation in an optimal basic solution.
- MSK_DPAR_BASIS_TOL_X. Maximum absolute primal bound violation allowed in an optimal basic solution.

- MSK_IPAR_BI_MAX_ITERATIONS. Maximum number of iterations after basis identification.
- MSK_DPAR_INTPNT_CO_TOL_DFEAS. Dual feasibility tolerance used by the conic interior-point optimizer.
- MSK_DPAR_INTPNT_CO_TOL_INFEAS. Infeasibility tolerance for the conic solver.
- MSK_DPAR_INTPNT_CO_TOL_MU_RED. Optimality tolerance for the conic solver.
- MSK_DPAR_INTPNT_CO_TOL_NEAR_REL. Optimality tolerance for the conic solver.
- MSK_DPAR_INTPNT_CO_TOL_PFEAS. Primal feasibility tolerance used by the conic interior-point optimizer.
- MSK_DPAR_INTPNT_CO_TOL_REL_GAP. Relative gap termination tolerance used by the conic interior-point optimizer.
- MSK_IPAR_INTPNT_MAX_ITERATIONS. Controls the maximum number of iterations allowed in the interior-point optimizer.
- MSK_DPAR_INTPNT_NL_TOL_DFEAS. Dual feasibility tolerance used when a nonlinear model is solved.
- MSK_DPAR_INTPNT_NL_TOL_MU_RED. Relative complementarity gap tolerance.
- MSK_DPAR_INTPNT_NL_TOL_NEAR_REL. Nonlinear solver optimality tolerance parameter.
- MSK_DPAR_INTPNT_NL_TOL_PFEAS. Primal feasibility tolerance used when a nonlinear model is solved.
- MSK_DPAR_INTPNT_NL_TOL_REL_GAP. Relative gap termination tolerance for nonlinear problems.
- MSK_DPAR_INTPNT_TOL_DFEAS. Dual feasibility tolerance used for linear and quadratic optimization problems.
- MSK_DPAR_INTPNT_TOL_INFEAS. Nonlinear solver infeasibility tolerance parameter.
- MSK_DPAR_INTPNT_TOL_MU_RED. Relative complementarity gap tolerance.
- MSK_DPAR_INTPNT_TOL_PFEAS. Primal feasibility tolerance used for linear and quadratic optimization problems.
- MSK_DPAR_INTPNT_TOL_REL_GAP. Relative gap termination tolerance.
- MSK_DPAR_LOWER_OBJ_CUT. Objective bound.
- MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH. Objective bound.
- MSK_DPAR_MIO_DISABLE_TERM_TIME. Certain termination criteria is disabled within the mixed-integer optimizer for period time specified by the parameter.
- MSK_IPAR_MIO_MAX_NUM_BRANCHES. Maximum number of branches allowed during the branch and bound search.
- MSK_IPAR_MIO_MAX_NUM_SOLUTIONS. Controls how many feasible solutions the mixed-integer optimizer investigates.
- MSK_DPAR_MIO_MAX_TIME. Time limit for the mixed-integer optimizer.
- MSK_DPAR_MIO_NEAR_TOL_REL_GAP. The mixed-integer optimizer is terminated when this tolerance is satisfied.

  - MSK_DPAR_MIO_REL_GAP_CONST. This value is used to compute the relative gap for the solution to an integer optimization problem.
  - MSK_DPAR_MIO_TOL_REL_GAP. Relative optimality tolerance employed by the mixed-integer optimizer.
  - MSK_DPAR_OPTIMIZER_MAX_TIME. Solver time limit.
  - MSK_IPAR_SIM_MAX_ITERATIONS. Maximum number of iterations that can be used by a simplex optimizer.
  - MSK_DPAR_UPPER_OBJ_CUT. Objective bound.
  - MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH. Objective bound.

- Integer parameters
- Double parameters
- String parameters

9.1 MSKdparame: Double parameters

9.1.1 MSK_DPAR_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-6`

9.1.2 MSK_DPAR_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`

9.1.3 MSK_DPAR_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`

9.1.4 MSK_DPAR_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`

9.1.5 MSK_DPAR_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`

9.1.6 MSK_DPAR_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`

9.1.7 MSK_DPAR_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`

9.1.8 MSK_DPAR_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`

9.1.9 MSK_DPAR_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`

9.1.10 MSK_DPAR_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`

9.1.11 MSK_DPAR_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`

9.1.12 MSK_DPAR_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`

9.1.13 MSK_DPAR_DATA_TOL_QIJ

Corresponding constant:

MSK_DPAR_DATA_TOL_QIJ

Description:

Absolute zero tolerance for elements in Q matrixes.

Possible Values:

Any number between 0.0 and +inf.

Default value:

`1.0e-16`

9.1.14 MSK_DPAR_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`

9.1.15 MSK_DPAR_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`

9.1.16 MSK_DPAR_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.

9.1.17 MSK_DPAR_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-10`

9.1.18 MSK_DPAR_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`

9.1.19 MSK_DPAR_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:

`1000`

9.1.20 MSK_DPAR_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.

9.1.21 MSK_DPAR_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-7`

See also:

- `MSK_DPAR_INTPNT_CO_TOL_NEAR_REL` Optimality tolerance for the conic solver.

9.1.22 MSK_DPAR_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`

9.1.23 MSK_DPAR_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`

9.1.24 MSK_DPAR_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`

9.1.25 MSK_DPAR_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`

9.1.26 MSK_DPAR_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`

9.1.27 MSK_DPAR_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`

9.1.28 MSK_DPAR_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`

9.1.29 MSK_DPAR_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`

9.1.30 MSK_DPAR_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`

9.1.31 MSK_DPAR_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-10`

9.1.32 MSK_DPAR_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`

9.1.33 MSK_DPAR_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`

9.1.34 MSK_DPAR_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`

9.1.35 MSK_DPAR_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`

9.1.36 MSK_DPAR_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`

9.1.37 MSK_DPAR_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`

9.1.38 MSK_DPAR_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. In other words the interior-point optimizer does not make any progress and therefore it is better stop.

Possible Values:

Any number between 0.0 and 1.0.

Default value:

`1.0e-6`

9.1.39 MSK_DPAR_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.

9.1.40 MSK_DPAR_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`

9.1.41 MSK_DPAR_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.

9.1.42 MSK_DPAR_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`

9.1.43 MSK_DPAR_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`

9.1.44 MSK_DPAR_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`

9.1.45 MSK_DPAR_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.

9.1.46 MSK_DPAR_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.

9.1.47 MSK_DPAR_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`

9.1.48 MSK_DPAR_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`

9.1.49 MSK_DPAR_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`

9.1.50 MSK_DPAR_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 1e-9 and +inf.

Default value:

`1.0e-5`

9.1.51 MSK_DPAR_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`

9.1.52 MSK_DPAR_MIO_TOL_MAX_CUT_FRAC_RHS

Corresponding constant:

MSK_DPAR_MIO_TOL_MAX_CUT_FRAC_RHS

Description:

Maximum value of fractional part of right hand side to generate CMIR and CG cuts for. A value of 0.0 means that the value is selected automatically.

Possible Values:

Any number between 0.0 and 1.0.

Default value:

`0.0`

9.1.53 MSK_DPAR_MIO_TOL_MIN_CUT_FRAC_RHS

Corresponding constant:

MSK_DPAR_MIO_TOL_MIN_CUT_FRAC_RHS

Description:

Minimum value of fractional part of right hand side to generate CMIR and CG cuts for. A value of 0.0 means that the value is selected automatically.

Possible Values:

Any number between 0.0 and 1.0.

Default value:

`0.0`

9.1.54 MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT

Corresponding constant:

MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT

Description:

If the relative improvement of the dual bound is smaller than this value, the solver will terminate the root cut generation. A value of 0.0 means that the value is selected automatically.

Possible Values:

Any number between 0.0 and 1.0.

Default value:

`0.0`

9.1.55 MSK_DPAR_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`

9.1.56 MSK_DPAR_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`

9.1.57 MSK_DPAR_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`

9.1.58 MSK_DPAR_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`

9.1.59 MSK_DPAR_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`

9.1.60 MSK_DPAR_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`

9.1.61 MSK_DPAR_PRESOLVE_TOL_ABS_LINDEP

Corresponding constant:

MSK_DPAR_PRESOLVE_TOL_ABS_LINDEP

Description:

Absolute tolerance employed by the linear dependency checker.

Possible Values:

Any number between 0.0 and +inf.

Default value:

`1.0e-6`

9.1.62 MSK_DPAR_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`

9.1.63 MSK_DPAR_PRESOLVE_TOL_REL_LINDEP

Corresponding constant:

MSK_DPAR_PRESOLVE_TOL_REL_LINDEP

Description:

Relative tolerance employed by the linear dependency checker.

Possible Values:

Any number between 0.0 and +inf.

Default value:

`1.0e-10`

9.1.64 MSK_DPAR_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`

9.1.65 MSK_DPAR_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`

9.1.66 MSK_DPAR_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`

9.1.67 MSK_DPAR_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`

9.1.68 MSK_DPAR_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`

9.1.69 MSK_DPAR_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.

9.1.70 MSK_DPAR_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`

9.2 MSKiparame: Integer parameters

9.2.1 MSK_IPAR_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`

9.2.2 MSK_IPAR_ANA_SOL_BASIS

Corresponding constant:

MSK_IPAR_ANA_SOL_BASIS

Description:

Controls whether the basis matrix is analyzed in solaution analyzer.

Possible values:

- `MSK_OFF` Switch the option off.
- `MSK_ON` Switch the option on.

Default value:

`MSK_ON`

9.2.3 MSK_IPAR_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_OFF` Switch the option off.

- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.4 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.5 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.6 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.7 MSK_IPAR_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_CONCURRENT The optimizer for nonconvex nonlinear problems.
- MSK_OPTIMIZER_CONIC The optimizer for problems having conic constraints.
- MSK_OPTIMIZER_DUAL_SIMPLEX The dual simplex optimizer is used.
- MSK_OPTIMIZER_FREE The optimizer is chosen automatically.
- MSK_OPTIMIZER_FREE_SIMPLEX One of the simplex optimizers is used.
- MSK_OPTIMIZER_INTPNT The interior-point optimizer is used.
- MSK_OPTIMIZER_MIXED_INT The mixed-integer optimizer.
- MSK_OPTIMIZER_MIXED_INT_CONIC The mixed-integer optimizer for conic and linear problems.
- MSK_OPTIMIZER_NETWORK_PRIMAL_SIMPLEX The network primal simplex optimizer is used. It is only applicable to pute network problems.
- MSK_OPTIMIZER_NONCONVEX The optimizer for nonconvex nonlinear problems.
- MSK_OPTIMIZER_PRIMAL_DUAL_SIMPLEX The primal dual simplex optimizer is used.
- MSK_OPTIMIZER_PRIMAL_SIMPLEX The primal simplex optimizer is used.

Default value:

MSK_OPTIMIZER_FREE

9.2.8 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.9 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.10 MSK_IPAR_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`

9.2.11 MSK_IPAR_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_OFF` Switch the option off.
- `MSK_ON` Switch the option on.

Default value:

`MSK_ON`

9.2.12 MSK_IPAR_CHECK_CONVEXITY

Corresponding constant:

MSK_IPAR_CHECK_CONVEXITY

Description:

Specify the level of convexity check on quadratic problems

Possible values:

- `MSK_CHECK_CONVEXITY_FULL` Perform a full convexity check.
- `MSK_CHECK_CONVEXITY_NONE` No convexity check.
- `MSK_CHECK_CONVEXITY_SIMPLE` Perform simple and fast convexity check.

Default value:

`MSK_CHECK_CONVEXITY_FULL`

9.2.13 MSK_IPAR_COMPRESS_STATFILE

Corresponding constant:

MSK_IPAR_COMPRESS_STATFILE

Description:

Control compression of stat files.

Possible values:

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.14 MSK_IPAR_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

9.2.15 MSK_IPAR_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

9.2.16 MSK_IPAR_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`

9.2.17 MSK_IPAR_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`

9.2.18 MSK_IPAR_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`

9.2.19 MSK_IPAR_FEASREPAIR_OPTIMIZE

Corresponding constant:

MSK_IPAR_FEASREPAIR_OPTIMIZE

Description:

Controls which type of feasibility analysis is to be performed.

Possible values:

- MSK_FEASREPAIR_OPTIMIZE_COMBINED Minimize with original objective subject to minimal weighted violation of bounds.
- MSK_FEASREPAIR_OPTIMIZE_NONE Do not optimize the feasibility repair problem.
- MSK_FEASREPAIR_OPTIMIZE_PENALTY Minimize weighted sum of violations.

Default value:

MSK_FEASREPAIR_OPTIMIZE_NONE

9.2.20 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.21 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.22 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.23 MSK_IPAR_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

9.2.24 MSK_IPAR_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_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_NEVER Never do basis identification.
- MSK_BI_NO_ERROR Basis identification is performed if the interior-point optimizer terminates without an error.
- MSK_BI_RESERVERED Not currently in use.

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.

9.2.25 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.26 MSK_IPAR_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`

9.2.27 MSK_IPAR_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`

9.2.28 MSK_IPAR_INTPNT_HOTSTART

Corresponding constant:

MSK_IPAR_INTPNT_HOTSTART

Description:

Currently not in use.

Possible values:

- `MSK_INTPNT_HOTSTART_DUAL` The interior-point optimizer exploits the dual solution only.
- `MSK_INTPNT_HOTSTART_NONE` The interior-point optimizer performs a coldstart.
- `MSK_INTPNT_HOTSTART_PRIMAL` The interior-point optimizer exploits the primal solution only.
- `MSK_INTPNT_HOTSTART_PRIMAL_DUAL` The interior-point optimizer exploits both the primal and dual solution.

Default value:

MSK_INTPNT_HOTSTART_NONE

9.2.29 MSK_IPAR_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

9.2.30 MSK_IPAR_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

9.2.31 MSK_IPAR_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

9.2.32 MSK_IPAR_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

9.2.33 MSK_IPAR_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_APPMINLOC Approximate minimum local fill-in ordering is employed.
- MSK_ORDER_METHOD_EXPERIMENTAL This option should not be used.
- MSK_ORDER_METHOD_FORCE_GRAPHPAR Always use the graph partitioning based ordering even if it is worse that the approximate minimum local fill ordering.
- MSK_ORDER_METHOD_FREE The ordering method is chosen automatically.
- MSK_ORDER_METHOD_NONE No ordering is used.
- MSK_ORDER_METHOD_TRY_GRAPHPAR Always try the the graph partitioning based ordering.

Default value:

MSK_ORDER_METHOD_FREE

9.2.34 MSK_IPAR_INTPNT_REGULARIZATION_USE

Corresponding constant:

MSK_IPAR_INTPNT_REGULARIZATION_USE

Description:

Controls whether regularization is allowed.

Possible values:

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.35 MSK_IPAR_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_AGGRESSIVE A very aggressive scaling is performed.
- MSK_SCALING_FREE The optimizer chooses the scaling heuristic.
- MSK_SCALING_MODERATE A conservative scaling is performed.
- MSK_SCALING_NONE No scaling is performed.

Default value:

MSK_SCALING_FREE

9.2.36 MSK_IPAR_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_DUAL The optimizer should solve the dual problem.
- MSK_SOLVE_FREE The optimizer is free to solve either the primal or the dual problem.
- MSK_SOLVE_PRIMAL The optimizer should solve the primal problem.

Default value:

MSK_SOLVE_FREE

9.2.37 MSK_IPAR_INTPNT_STARTING_POINT

Corresponding constant:

MSK_IPAR_INTPNT_STARTING_POINT

Description:

Starting point used by the interior-point optimizer.

Possible values:

- MSK_STARTING_POINT_CONSTANT The 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.
- 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.

Default value:

MSK_STARTING_POINT_FREE

9.2.38 MSK_IPAR_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

9.2.39 MSK_IPAR_LICENSE_DEBUG

Corresponding constant:

MSK_IPAR_LICENSE_DEBUG

Description:

This option is used to turn on debugging of the incense manager.

Possible values:

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.40 MSK_IPAR_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

9.2.41 MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS

Corresponding constant:

MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS

Description:

Controls whether license features expire warnings are suppressed.

Possible values:

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.42 MSK_IPAR_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_OFF` Switch the option off.
- `MSK_ON` Switch the option on.

Default value:

`MSK_OFF`

9.2.43 MSK_IPAR_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.

9.2.44 MSK_IPAR_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

9.2.45 MSK_IPAR_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

9.2.46 MSK_IPAR_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

9.2.47 MSK_IPAR_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

9.2.48 MSK_IPAR_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.

9.2.49 MSK_IPAR_LOG_EXPAND

Corresponding constant:

MSK_IPAR_LOG_EXPAND

Description:

Controls the amount of logging when a data item such as the maximum number constrains is expanded.

Possible Values:

Any number between 0 and +inf.

Default value:

0

9.2.50 MSK_IPAR_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

9.2.51 MSK_IPAR_LOG_FEAS_REPAIR

Corresponding constant:

MSK_IPAR_LOG_FEAS_REPAIR

Description:

Controls the amount of output printed when performing feasibility repair. A value higher than one means extensive logging.

Possible Values:

Any number between 0 and +inf.

Default value:

1

9.2.52 MSK_IPAR_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

9.2.53 MSK_IPAR_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

9.2.54 MSK_IPAR_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

9.2.55 MSK_IPAR_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

9.2.56 MSK_IPAR_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

9.2.57 MSK_IPAR_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

9.2.58 MSK_IPAR_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

9.2.59 MSK_IPAR_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

9.2.60 MSK_IPAR_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

9.2.61 MSK_IPAR_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

9.2.62 MSK_IPAR_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

9.2.63 MSK_IPAR_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

9.2.64 MSK_IPAR_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`

9.2.65 MSK_IPAR_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`

9.2.66 MSK_IPAR_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`

9.2.67 MSK_IPAR_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:

`1000`

9.2.68 MSK_IPAR_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`

9.2.69 MSK_IPAR_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:

`1000`

9.2.70 MSK_IPAR_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

9.2.71 MSK_IPAR_MAX_NUM_WARNINGS

Corresponding constant:

MSK_IPAR_MAX_NUM_WARNINGS

Description:

A negtive number means all warnings are logged. Otherwise the parameter specifies the maximum number times each warning is logged.

Possible Values:

Any number between -inf and +inf.

Default value:

6

9.2.72 MSK_IPAR_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_FREE The mixed-integer optimizer decides which branch to choose.
- MSK_BRANCH_DIR_UP The mixed-integer optimizer always chooses the up branch first.

Default value:

MSK_BRANCH_DIR_FREE

9.2.73 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.74 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.75 MSK_IPAR_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_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.
- 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.

Default value:

MSK_MIO_CONT_SOL_NONE

9.2.76 MSK_IPAR_MIO_CUT_CG

Corresponding constant:

MSK_IPAR_MIO_CUT_CG

Description:

Controls whether CG cuts should be generated.

Possible values:

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.77 MSK_IPAR_MIO_CUT_CMIR

Corresponding constant:

MSK_IPAR_MIO_CUT_CMIR

Description:

Controls whether mixed integer rounding cuts should be generated.

Possible values:

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.78 MSK_IPAR_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`

9.2.79 MSK_IPAR_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`

9.2.80 MSK_IPAR_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`

9.2.81 MSK_IPAR_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`

9.2.82 MSK_IPAR_MIO_HOTSTART

Corresponding constant:

MSK_IPAR_MIO_HOTSTART

Description:

Controls whether the integer optimizer is hot-started.

Possible values:

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.83 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.84 MSK_IPAR_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

9.2.85 MSK_IPAR_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.

9.2.86 MSK_IPAR_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.

9.2.87 MSK_IPAR_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.

9.2.88 MSK_IPAR_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`

9.2.89 MSK_IPAR_MIO_MT_USER_CB

Corresponding constant:

MSK_IPAR_MIO_MT_USER_CB

Description:

It true user callbacks are called from each thread used by this optimizer. If false the user callback is only called from a single thread.

Possible values:

- `MSK_OFF` Switch the option off.
- `MSK_ON` Switch the option on.

Default value:

`MSK_ON`

9.2.90 MSK_IPAR_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_CONCURRENT The optimizer for nonconvex nonlinear problems.
- MSK_OPTIMIZER_CONIC The optimizer for problems having conic constraints.
- MSK_OPTIMIZER_DUAL_SIMPLEX The dual simplex optimizer is used.
- MSK_OPTIMIZER_FREE The optimizer is chosen automatically.
- MSK_OPTIMIZER_FREE_SIMPLEX One of the simplex optimizers is used.
- MSK_OPTIMIZER_INTPNT The interior-point optimizer is used.
- MSK_OPTIMIZER_MIXED_INT The mixed-integer optimizer.
- MSK_OPTIMIZER_MIXED_INT_CONIC The mixed-integer optimizer for conic and linear problems.
- MSK_OPTIMIZER_NETWORK_PRIMAL_SIMPLEX The network primal simplex optimizer is used. It is only applicable to pute network problems.
- MSK_OPTIMIZER_NONCONVEX The optimizer for nonconvex nonlinear problems.
- MSK_OPTIMIZER_PRIMAL_DUAL_SIMPLEX The primal dual simplex optimizer is used.
- MSK_OPTIMIZER_PRIMAL_SIMPLEX The primal simplex optimizer is used.

Default value:

MSK_OPTIMIZER_FREE

9.2.91 MSK_IPAR_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_BEST The optimizer employs a best bound node selection strategy.
- MSK_MIO_NODE_SELECTION_FIRST The optimizer employs a depth first node selection strategy.
- MSK_MIO_NODE_SELECTION_FREE The optimizer decides the node selection strategy.

- MSK_MIO_NODE_SELECTION_HYBRID The optimizer employs a hybrid strategy.
- MSK_MIO_NODE_SELECTION_PSEUDO The optimizer employs selects the node based on a pseudo cost estimate.
- MSK_MIO_NODE_SELECTION_WORST The optimizer employs a worst bound node selection strategy.

Default value:

MSK_MIO_NODE_SELECTION_FREE

9.2.92 MSK_IPAR_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

9.2.93 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.94 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.95 MSK_IPAR_MIO_PRESOLVE_USE

Corresponding constant:

MSK_IPAR_MIO_PRESOLVE_USE

Description:

Controls whether presolve is performed by the mixed-integer optimizer.

Possible values:

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.96 MSK_IPAR_MIO_PROBING_LEVEL

Corresponding constant:

MSK_IPAR_MIO_PROBING_LEVEL

Description:

Controls the amount of probing employed by the mixed-integer optimizer in presolve.

- -1 The optimizer chooses the level of probing employed.
- 0 Probing is disabled.
- 1 A low amount of probing is employed.

- 2 A medium amount of probing is employed.
- 3 A high amount of probing is employed.

Possible Values:

An integer value in the range of -1 to 3.

Default value:

`-1`

9.2.97 MSK_IPAR_MIO_RINS_MAX_NODES

Corresponding constant:

MSK_IPAR_MIO_RINS_MAX_NODES

Description:

Controls the maximum number of nodes allowed in each call to the RINS heuristic. The default value of -1 means that the value is determined automatically. A value of zero turns off the heuristic.

Possible Values:

Any number between -1 and +inf.

Default value:

`-1`

9.2.98 MSK_IPAR_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_CONCURRENT` The optimizer for nonconvex nonlinear problems.
- `MSK_OPTIMIZER_CONIC` The optimizer for problems having conic constraints.
- `MSK_OPTIMIZER_DUAL_SIMPLEX` The dual simplex optimizer is used.
- `MSK_OPTIMIZER_FREE` The optimizer is chosen automatically.
- `MSK_OPTIMIZER_FREE_SIMPLEX` One of the simplex optimizers is used.
- `MSK_OPTIMIZER_INTPNT` The interior-point optimizer is used.
- `MSK_OPTIMIZER_MIXED_INT` The mixed-integer optimizer.

- MSK_OPTIMIZER_MIXED_INT_CONIC The mixed-integer optimizer for conic and linear problems.
- MSK_OPTIMIZER_NETWORK_PRIMAL_SIMPLEX The network primal simplex optimizer is used. It is only applicable to pute network problems.
- MSK_OPTIMIZER_NONCONVEX The optimizer for nonconvex nonlinear problems.
- MSK_OPTIMIZER_PRIMAL_DUAL_SIMPLEX The primal dual simplex optimizer is used.
- MSK_OPTIMIZER_PRIMAL_SIMPLEX The primal simplex optimizer is used.

Default value:

MSK_OPTIMIZER_FREE

9.2.99 MSK_IPAR_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

9.2.100 MSK_IPAR_MIO_USE_MULTITHREADED_OPTIMIZER

Corresponding constant:

MSK_IPAR_MIO_USE_MULTITHREADED_OPTIMIZER

Description:

Controls wheter the new multithreaded optimizer should be used for Mixed integer problems.

Possible values:

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.101 MSK_IPAR_MT_SPINCOUNT

Corresponding constant:

MSK_IPAR_MT_SPINCOUNT

Description:

Set the number of iterations to spin before sleeping.

Possible Values:

Any integer greater or equal to 0.

Default value:

`0`

9.2.102 MSK_IPAR_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`

9.2.103 MSK_IPAR_NUM_THREADS

Corresponding constant:

MSK_IPAR_NUM_THREADS

Description:

Controls the number of threads employed by the optimizer. If set to 0 the number of threads used will be equal to the number of cores detected on the machine.

Possible Values:

Any integer greater or equal to 0.

Default value:

`0`

9.2.104 MSK_IPAR_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

9.2.105 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.106 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.107 MSK_IPAR_OPF_WRITE_PARAMETERS

Corresponding constant:

MSK_IPAR_OPF_WRITE_PARAMETERS

Description:

Write a parameter section in an OPF file.

Possible values:

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.108 MSK_IPAR_OPF_WRITE_PROBLEM

Corresponding constant:

MSK_IPAR_OPF_WRITE_PROBLEM

Description:

Write objective, constraints, bounds etc. to an OPF file.

Possible values:

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.109 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.110 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.111 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.112 MSK_IPAR_OPF_WRITE_SOLUTIONS

Corresponding constant:

MSK_IPAR_OPF_WRITE_SOLUTIONS

Description:

Enable inclusion of solutions in the OPF files.

Possible values:

- MSK_OFF Switch the option off.

- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.113 MSK_IPAR_OPTIMIZER

Corresponding constant:

MSK_IPAR_OPTIMIZER

Description:

The paramter controls which optimizer is used to optimize the task.

Possible values:

- MSK_OPTIMIZER_CONCURRENT The optimizer for nonconvex nonlinear problems.
- MSK_OPTIMIZER_CONIC The optimizer for problems having conic constraints.
- MSK_OPTIMIZER_DUAL_SIMPLEX The dual simplex optimizer is used.
- MSK_OPTIMIZER_FREE The optimizer is chosen automatically.
- MSK_OPTIMIZER_FREE_SIMPLEX One of the simplex optimizers is used.
- MSK_OPTIMIZER_INTPNT The interior-point optimizer is used.
- MSK_OPTIMIZER_MIXED_INT The mixed-integer optimizer.
- MSK_OPTIMIZER_MIXED_INT_CONIC The mixed-integer optimizer for conic and linear problems.
- MSK_OPTIMIZER_NETWORK_PRIMAL_SIMPLEX The network primal simplex optimizer is used. It is only applicable to pute network problems.
- MSK_OPTIMIZER_NONCONVEX The optimizer for nonconvex nonlinear problems.
- MSK_OPTIMIZER_PRIMAL_DUAL_SIMPLEX The primal dual simplex optimizer is used.
- MSK_OPTIMIZER_PRIMAL_SIMPLEX The primal simplex optimizer is used.

Default value:

MSK_OPTIMIZER_FREE

9.2.114 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.115 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.116 MSK_IPAR_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

9.2.117 MSK_IPAR_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

9.2.118 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.119 MSK_IPAR_PRESOLVE_LEVEL

Corresponding constant:

MSK_IPAR_PRESOLVE_LEVEL

Description:

Currently not used.

Possible Values:

Any number between -inf and +inf.

Default value:

-1

9.2.120 MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH

Corresponding constant:

MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH

Description:

The linear dependency check is potentially computationally expensive.

Possible Values:

Any number between 0 and +inf.

Default value:

`100`

9.2.121 MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH

Corresponding constant:

MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH

Description:

The linear dependency check is potentially computationally expensive.

Possible Values:

Any number between 0 and +inf.

Default value:

`100`

9.2.122 MSK_IPAR_PRESOLVE_LINDEP_USE

Corresponding constant:

MSK_IPAR_PRESOLVE_LINDEP_USE

Description:

Controls whether the linear constraints are checked for linear dependencies.

Possible values:

- `MSK_OFF` Switch the option off.
- `MSK_ON` Switch the option on.

Default value:

`MSK_ON`

9.2.123 MSK_IPAR_PRESOLVE_MAX_NUM_REDUCTIONS

Corresponding constant:

MSK_IPAR_PRESOLVE_MAX_NUM_REDUCTIONS

Description:

Controls the maximum number reductions performed by the presolve. The value of the parameter is normally only changed in connection with debugging. A negative value implies that an infinite number of reductions are allowed.

Possible Values:

Any number between -inf and +inf.

Default value:

`-1`

9.2.124 MSK_IPAR_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_FREE` It is decided automatically whether to presolve before the problem is optimized.
- `MSK_PRESOLVE_MODE_OFF` The problem is not presolved before it is optimized.
- `MSK_PRESOLVE_MODE_ON` The problem is presolved before it is optimized.

Default value:

`MSK_PRESOLVE_MODE_FREE`

9.2.125 MSK_IPAR_PRIMAL_REPAIR_OPTIMIZER

Corresponding constant:

MSK_IPAR_PRIMAL_REPAIR_OPTIMIZER

Description:

Controls which optimizer that is used to find the optimal repair.

Possible values:

- `MSK_OPTIMIZER_CONCURRENT` The optimizer for nonconvex nonlinear problems.

- MSK_OPTIMIZER_CONIC The optimizer for problems having conic constraints.
- MSK_OPTIMIZER_DUAL_SIMPLEX The dual simplex optimizer is used.
- MSK_OPTIMIZER_FREE The optimizer is chosen automatically.
- MSK_OPTIMIZER_FREE_SIMPLEX One of the simplex optimizers is used.
- MSK_OPTIMIZER_INTPNT The interior-point optimizer is used.
- MSK_OPTIMIZER_MIXED_INT The mixed-integer optimizer.
- MSK_OPTIMIZER_MIXED_INT_CONIC The mixed-integer optimizer for conic and linear problems.
- MSK_OPTIMIZER_NETWORK_PRIMAL_SIMPLEX The network primal simplex optimizer is used. It is only applicable to pute network problems.
- MSK_OPTIMIZER_NONCONVEX The optimizer for nonconvex nonlinear problems.
- MSK_OPTIMIZER_PRIMAL_DUAL_SIMPLEX The primal dual simplex optimizer is used.
- MSK_OPTIMIZER_PRIMAL_SIMPLEX The primal simplex optimizer is used.

Default value:

MSK_OPTIMIZER_FREE

9.2.126 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.127 MSK_IPAR_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`

9.2.128 MSK_IPAR_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`

9.2.129 MSK_IPAR_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`

9.2.130 MSK_IPAR_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_COMPRESS_FREE The type of compression used is chosen automatically.
- MSK_COMPRESS_GZIP The type of compression used is gzip compatible.
- MSK_COMPRESS_NONE No compression is used.

Default value:

MSK_COMPRESS_FREE

9.2.131 MSK_IPAR_READ_DATA_FORMAT

Corresponding constant:

MSK_IPAR_READ_DATA_FORMAT

Description:

Format of the data file to be read.

Possible values:

- MSK_DATA_FORMAT_CB Conic benchmark format.
- MSK_DATA_FORMAT_EXTENSION The file extension is used to determine the data file format.
- MSK_DATA_FORMAT_FREE_MPS The data data a free MPS formatted file.
- MSK_DATA_FORMAT_LP The data file is LP formatted.
- MSK_DATA_FORMAT_MPS The data file is MPS formatted.
- MSK_DATA_FORMAT_OP The data file is an optimization problem formatted file.
- MSK_DATA_FORMAT_TASK Generic task dump file.
- MSK_DATA_FORMAT_XML The data file is an XML formatted file.

Default value:

MSK_DATA_FORMAT_EXTENSION

9.2.132 MSK_IPAR_READ_DEBUG

Corresponding constant:

MSK_IPAR_READ_DEBUG

Description:

Turns on additional debugging information when reading files.

Possible values:

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.133 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.134 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.135 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.136 MSK_IPAR_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_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.
- MSK_MPS_FORMAT_RELAXED It is assumed that the input file satisfies a slightly relaxed version of the MPS format.
- MSK_MPS_FORMAT_STRICT It is assumed that the input file satisfies the MPS format strictly.

Default value:

MSK_MPS_FORMAT_RELAXED

9.2.137 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.138 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.139 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.140 MSK_IPAR_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

9.2.141 MSK_IPAR_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`

9.2.142 MSK_IPAR_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_OFF` Switch the option off.
- `MSK_ON` Switch the option on.

Default value:

`MSK_OFF`

9.2.143 MSK_IPAR_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`

9.2.144 MSK_IPAR_SENSITIVITY_ALL

Corresponding constant:

MSK_IPAR_SENSITIVITY_ALL

Description:

Not applicable.

Possible values:

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.145 MSK_IPAR_SENSITIVITY_OPTIMIZER

Corresponding constant:

MSK_IPAR_SENSITIVITY_OPTIMIZER

Description:

Controls which optimizer is used for optimal partition sensitivity analysis.

Possible values:

- MSK_OPTIMIZER_CONCURRENT The optimizer for nonconvex nonlinear problems.
- MSK_OPTIMIZER_CONIC The optimizer for problems having conic constraints.
- MSK_OPTIMIZER_DUAL_SIMPLEX The dual simplex optimizer is used.
- MSK_OPTIMIZER_FREE The optimizer is chosen automatically.
- MSK_OPTIMIZER_FREE_SIMPLEX One of the simplex optimizers is used.
- MSK_OPTIMIZER_INTPNT The interior-point optimizer is used.
- MSK_OPTIMIZER_MIXED_INT The mixed-integer optimizer.
- MSK_OPTIMIZER_MIXED_INT_CONIC The mixed-integer optimizer for conic and linear problems.
- MSK_OPTIMIZER_NETWORK_PRIMAL_SIMPLEX The network primal simplex optimizer is used. It is only applicable to pute network problems.
- MSK_OPTIMIZER_NONCONVEX The optimizer for nonconvex nonlinear problems.
- MSK_OPTIMIZER_PRIMAL_DUAL_SIMPLEX The primal dual simplex optimizer is used.
- MSK_OPTIMIZER_PRIMAL_SIMPLEX The primal simplex optimizer is used.

Default value:

MSK_OPTIMIZER_FREE_SIMPLEX

9.2.146 MSK_IPAR_SENSITIVITY_TYPE

Corresponding constant:

MSK_IPAR_SENSITIVITY_TYPE

Description:

Controls which type of sensitivity analysis is to be performed.

Possible values:

- MSK_SENSITIVITY_TYPE_BASIS Basis sensitivity analysis is performed.
- MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION Optimal partition sensitivity analysis is performed.

Default value:

MSK_SENSITIVITY_TYPE_BASIS

9.2.147 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.148 MSK_IPAR_SIM_DEGEN

Corresponding constant:

MSK_IPAR_SIM_DEGEN

Description:

Controls how aggressively degeneration is handled.

Possible values:

- MSK_SIM_DEGEN_AGGRESSIVE The simplex optimizer should use an aggressive degeneration strategy.
- MSK_SIM_DEGEN_FREE The simplex optimizer chooses the degeneration strategy.
- MSK_SIM_DEGEN_MINIMUM The simplex optimizer should use a minimum degeneration strategy.
- MSK_SIM_DEGEN_MODERATE The simplex optimizer should use a moderate degeneration strategy.
- MSK_SIM_DEGEN_NONE The simplex optimizer should use no degeneration strategy.

Default value:

MSK_SIM_DEGEN_FREE

9.2.149 MSK_IPAR_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

9.2.150 MSK_IPAR_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

9.2.151 MSK_IPAR_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`

9.2.152 MSK_IPAR_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_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_FREE` The optimizer chooses the pricing strategy.
- `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_SE` The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

Default value:

`MSK_SIM_SELECTION_FREE`

9.2.153 MSK_IPAR_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_FREE The simplex optimizer can choose freely.
- MSK_SIM_EXPLOIT_DUPVEC_OFF Disallow the simplex optimizer to exploit duplicated columns.
- MSK_SIM_EXPLOIT_DUPVEC_ON Allow the simplex optimizer to exploit duplicated columns.

Default value:

MSK_SIM_EXPLOIT_DUPVEC_OFF

9.2.154 MSK_IPAR_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_FREE The simplex optimize chooses the hot-start type.
- 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.

Default value:

MSK_SIM_HOTSTART_FREE

9.2.155 MSK_IPAR_SIM_HOTSTART_LU

Corresponding constant:

MSK_IPAR_SIM_HOTSTART_LU

Description:

Determines if the simplex optimizer should exploit the initial factorization.

Possible values:

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.156 MSK_IPAR_SIM_INTEGER

Corresponding constant:

MSK_IPAR_SIM_INTEGER

Description:

An exprimental feature.

Possible Values:

Any number between 0 and 10.

Default value:

0

9.2.157 MSK_IPAR_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

9.2.158 MSK_IPAR_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

9.2.159 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.160 MSK_IPAR_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

9.2.161 MSK_IPAR_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

9.2.162 MSK_IPAR_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

9.2.163 MSK_IPAR_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_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_FREE The optimizer chooses the pricing strategy.
- 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_SE The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

Default value:

MSK_SIM_SELECTION_FREE

9.2.164 MSK_IPAR_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

9.2.165 MSK_IPAR_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_AGGRESSIVE The simplex optimizer should use an aggressive reformulation strategy.
- MSK_SIM_REFORMULATION_FREE The simplex optimizer can choose freely.
- MSK_SIM_REFORMULATION_OFF Disallow the simplex optimizer to reformulate the problem.

- MSK_SIM_REFORMULATION_ON Allow the simplex optimizer to reformulate the problem.

Default value:

MSK_SIM_REFORMULATION_OFF

9.2.166 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.167 MSK_IPAR_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_AGGRESSIVE A very aggressive scaling is performed.
- MSK_SCALING_FREE The optimizer chooses the scaling heuristic.
- MSK_SCALING_MODERATE A conservative scaling is performed.
- MSK_SCALING_NONE No scaling is performed.

Default value:

MSK_SCALING_FREE

9.2.168 MSK_IPAR_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_FREE` The optimizer chooses the scaling heuristic.
- `MSK_SCALING_METHOD_POW2` Scales only with power of 2 leaving the mantissa untouched.

Default value:

`MSK_SCALING_METHOD_POW2`

9.2.169 MSK_IPAR_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_DUAL` The optimizer should solve the dual problem.
- `MSK_SOLVE_FREE` The optimizer is free to solve either the primal or the dual problem.
- `MSK_SOLVE_PRIMAL` The optimizer should solve the primal problem.

Default value:

`MSK_SOLVE_FREE`

9.2.170 MSK_IPAR_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

9.2.171 MSK_IPAR_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_OFF` Switch the option off.
- `MSK_ON` Switch the option on.

Default value:

`MSK_OFF`

9.2.172 MSK_IPAR_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_OFF` Switch the option off.
- `MSK_ON` Switch the option on.

Default value:

`MSK_OFF`

9.2.173 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.174 MSK_IPAR_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

9.2.175 MSK_IPAR_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

9.2.176 MSK_IPAR_SOLUTION_CALLBACK

Corresponding constant:

MSK_IPAR_SOLUTION_CALLBACK

Description:

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

Possible values:

- `MSK_OFF` Switch the option off.
- `MSK_ON` Switch the option on.

Default value:

`MSK_OFF`

9.2.177 MSK_IPAR_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`

9.2.178 MSK_IPAR_WARNING_LEVEL

Corresponding constant:

MSK_IPAR_WARNING_LEVEL

Description:

Deprecated and not in use

Possible Values:

Any number between 0 and +inf.

Default value:

`1`

9.2.179 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.180 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.181 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.182 MSK_IPAR_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`

9.2.183 MSK_IPAR_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_CB` Conic benchmark format.
- `MSK_DATA_FORMAT_EXTENSION` The file extension is used to determine the data file format.
- `MSK_DATA_FORMAT_FREE_MPS` The data data a free MPS formatted file.
- `MSK_DATA_FORMAT_LP` The data file is LP formatted.
- `MSK_DATA_FORMAT_MPS` The data file is MPS formatted.
- `MSK_DATA_FORMAT_OP` The data file is an optimization problem formatted file.
- `MSK_DATA_FORMAT_TASK` Generic task dump file.
- `MSK_DATA_FORMAT_XML` The data file is an XML formatted file.

Default value:

`MSK_DATA_FORMAT_EXTENSION`

9.2.184 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.185 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.186 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.187 MSK_IPAR_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

9.2.188 MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_CONIC_ITEMS

Corresponding constant:

MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_CONIC_ITEMS

Description:

If the output format is not compatible with conic quadratic problems this parameter controls if the writer ignores the conic parts or produces an error.

Possible values:

- `MSK_OFF` Switch the option off.
- `MSK_ON` Switch the option on.

Default value:

`MSK_OFF`

9.2.189 MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_ITEMS

Corresponding constant:

MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_ITEMS

Description:

Controls if the writer ignores incompatible problem items when writing files.

Possible values:

- `MSK_OFF` Switch the option off.
- `MSK_ON` Switch the option on.

Default value:

`MSK_OFF`

9.2.190 MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_NL_ITEMS

Corresponding constant:

MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_NL_ITEMS

Description:

Controls if the writer ignores general non-linear terms or produces an error.

Possible values:

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.191 MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_PSD_ITEMS

Corresponding constant:

MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_PSD_ITEMS

Description:

If the output format is not compatible with semidefinite problems this parameter controls if the writer ignores the conic parts or produces an error.

Possible values:

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.192 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.193 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.194 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.195 MSK_IPAR_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

9.2.196 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.197 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.198 MSK_IPAR_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

9.2.199 MSK_IPAR_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_OFF` Switch the option off.
- `MSK_ON` Switch the option on.

Default value:

`MSK_ON`

9.2.200 MSK_IPAR_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`

9.2.201 MSK_IPAR_WRITE_SOL_BARVARIABLES

Corresponding constant:

MSK_IPAR_WRITE_SOL_BARVARIABLES

Description:

Controls whether the symmetric matrix variables section is written to the solution file.

Possible values:

- `MSK_OFF` Switch the option off.
- `MSK_ON` Switch the option on.

Default value:

`MSK_ON`

9.2.202 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.203 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.204 MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES

Corresponding constant:

MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES

Description:

Even if the names are invalid MPS names, then they are employed when writing the solution file.

Possible values:

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_OFF

9.2.205 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.206 MSK_IPAR_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_OFF Switch the option off.
- MSK_ON Switch the option on.

Default value:

MSK_ON

9.2.207 MSK_IPAR_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

9.3 MSKsparame: String parameter types

9.3.1 MSK_SPAR_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:

""

9.3.2 MSK_SPAR_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:

""

9.3.3 MSK_SPAR_DEBUG_FILE_NAME

Corresponding constant:

MSK_SPAR_DEBUG_FILE_NAME

Description:

MOSEK debug file.

Possible Values:

Any valid file name.

Default value:

""

9.3.4 MSK_SPAR_FEASREPAIR_NAME_PREFIX

Corresponding constant:

MSK_SPAR_FEASREPAIR_NAME_PREFIX

Description:

Not applicable.

Possible Values:

Any valid string.

Default value:

`"MSK-"`

9.3.5 MSK_SPAR_FEASREPAIR_NAME_SEPARATOR

Corresponding constant:

MSK_SPAR_FEASREPAIR_NAME_SEPARATOR

Description:

Not applicable.

Possible Values:

Any valid string.

Default value:

`"-"`

9.3.6 MSK_SPAR_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"`

9.3.7 MSK_SPAR_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:

`""`

9.3.8 MSK_SPAR_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:

`""`

9.3.9 MSK_SPAR_MIO_DEBUG_STRING

Corresponding constant:

MSK_SPAR_MIO_DEBUG_STRING

Description:

For internal use only.

Possible Values:

Any valid string.

Default value:

`""`

9.3.10 MSK_SPAR_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:

"%%"

9.3.11 MSK_SPAR_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:

""

9.3.12 MSK_SPAR_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:

""

9.3.13 MSK_SPAR_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:

""

9.3.14 MSK_SPAR_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:

""

9.3.15 MSK_SPAR_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:

""

9.3.16 MSK_SPAR_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:

""

9.3.17 MSK_SPAR_SENSITIVITY_FILE_NAME

Corresponding constant:

MSK_SPAR_SENSITIVITY_FILE_NAME

Description:

Not applicable.

Possible Values:

Any valid string.

Default value:

""

9.3.18 MSK_SPAR_SENSITIVITY_RES_FILE_NAME

Corresponding constant:

MSK_SPAR_SENSITIVITY_RES_FILE_NAME

Description:

Not applicable.

Possible Values:

Any valid string.

Default value:

""

9.3.19 MSK_SPAR_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:

`""`

9.3.20 MSK_SPAR_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:

`""`

9.3.21 MSK_SPAR_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:

""

9.3.22 MSK_SPAR_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:

""

9.3.23 MSK_SPAR_STAT_FILE_NAME

Corresponding constant:

MSK_SPAR_STAT_FILE_NAME

Description:

Statistics file name.

Possible Values:

Any valid file name.

Default value:

""

9.3.24 MSK_SPAR_STAT_KEY

Corresponding constant:

MSK_SPAR_STAT_KEY

Description:

Key used when writing the summary file.

Possible Values:

Any valid XML string.

Default value:

`""`

9.3.25 MSK_SPAR_STAT_NAME

Corresponding constant:

MSK_SPAR_STAT_NAME

Description:

Name used when writing the statistics file.

Possible Values:

Any valid XML string.

Default value:

`""`

9.3.26 MSK_SPAR_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"`

Chapter 10

Response codes

Response codes ordered by name.

`MSK_RES_ERR_AD_INVALID_CODELIST`

The code list data was invalid.

`MSK_RES_ERR_AD_INVALID_OPERAND`

The code list data was invalid. An unknown operand was used.

`MSK_RES_ERR_AD_INVALID_OPERATOR`

The code list data was invalid. An unknown operator was used.

`MSK_RES_ERR_AD_MISSING_OPERAND`

The code list data was invalid. Missing operand for operator.

`MSK_RES_ERR_AD_MISSING_RETURN`

The code list data was invalid. Missing return operation in function.

`MSK_RES_ERR_API_ARRAY_TOO_SMALL`

An input array was too short.

`MSK_RES_ERR_API_CB_CONNECT`

Failed to connect a callback object.

`MSK_RES_ERR_API_FATAL_ERROR`

An internal error occurred in the API. Please report this problem.

`MSK_RES_ERR_API_INTERNAL`

An internal fatal error occurred in an interface function.

`MSK_RES_ERR_ARG_IS_TOO_LARGE`

The value of a argument is too small.

MSK_RES_ERR_ARG_IS_TOO_SMALL
: The value of a argument is too small.

MSK_RES_ERR_ARGUMENT_DIMENSION
: A function argument is of incorrect dimension.

MSK_RES_ERR_ARGUMENT_IS_TOO_LARGE
: The value of a function argument is too large.

MSK_RES_ERR_ARGUMENT_LENNEQ
: Incorrect length of arguments.

MSK_RES_ERR_ARGUMENT_PERM_ARRAY
: An invalid permutation array is specified.

MSK_RES_ERR_ARGUMENT_TYPE
: Incorrect argument type.

MSK_RES_ERR_BAR_VAR_DIM
: The dimension of a symmetric matrix variable has to greater than 0.

MSK_RES_ERR_BASIS
: An invalid basis is specified. Either too many or too few basis variables are specified.

MSK_RES_ERR_BASIS_FACTOR
: The factorization of the basis is invalid.

MSK_RES_ERR_BASIS_SINGULAR
: The basis is singular and hence cannot be factored.

MSK_RES_ERR_BLANK_NAME
: An all blank name has been specified.

MSK_RES_ERR_CANNOT_CLONE_NL
: A task with a nonlinear function call-back cannot be cloned.

MSK_RES_ERR_CANNOT_HANDLE_NL
: A function cannot handle a task with nonlinear function call-backs.

MSK_RES_ERR_CBF_DUPLICATE_ACOORD
: Duplicate index in ACOORD.

MSK_RES_ERR_CBF_DUPLICATE_BCOORD
: Duplicate index in BCOORD.

MSK_RES_ERR_CBF_DUPLICATE_CON
: Duplicate CON keyword.

MSK_RES_ERR_CBF_DUPLICATE_INT
: Duplicate INT keyword.

MSK_RES_ERR_CBF_DUPLICATE_OBJ
: Duplicate OBJ keyword.

MSK_RES_ERR_CBF_DUPLICATE_OBJACOORD
: Duplicate index in OBJCOORD.

MSK_RES_ERR_CBF_DUPLICATE_VAR
: Duplicate VAR keyword.

MSK_RES_ERR_CBF_INVALID_CON_TYPE
: Invalid constraint type.

MSK_RES_ERR_CBF_INVALID_DOMAIN_DIMENSION
: Invalid domain dimension.

MSK_RES_ERR_CBF_INVALID_INT_INDEX
: Invalid INT index.

MSK_RES_ERR_CBF_INVALID_VAR_TYPE
: Invalid variable type.

MSK_RES_ERR_CBF_NO_VARIABLES
: No variables are specified.

MSK_RES_ERR_CBF_NO_VERSION_SPECIFIED
: No version specified.

MSK_RES_ERR_CBF_OBJ_SENSE
: An invalid objective sense is specified.

MSK_RES_ERR_CBF_PARSE
: An error occurred while parsing an CBF file.

MSK_RES_ERR_CBF_SYNTAX
: Invalid syntax.

MSK_RES_ERR_CBF_TOO_FEW_CONSTRAINTS
: Too few constraints defined.

MSK_RES_ERR_CBF_TOO_FEW_INTS
: Too few ints are specified.

MSK_RES_ERR_CBF_TOO_FEW_VARIABLES
: Too few variables defined.

MSK_RES_ERR_CBF_TOO_MANY_CONSTRAINTS

Too many constraints specified.

MSK_RES_ERR_CBF_TOO_MANY_INTS

Too many ints are specified.

MSK_RES_ERR_CBF_TOO_MANY_VARIABLES

Too many variables specified.

MSK_RES_ERR_CBF_UNSUPPORTED

Unsupported feature is present.

MSK_RES_ERR_CON_Q_NOT_NSD

The quadratic constraint matrix is not negative semidefinite as expected for a constraint with finite lower bound. This results in a nonconvex problem. The parameter MSK_DPAR_CHECK_CONVEXITY_REL_TOL can be used to relax the convexity check.

MSK_RES_ERR_CON_Q_NOT_PSD

The quadratic constraint matrix is not positive semidefinite as expected for a constraint with finite upper bound. This results in a nonconvex problem. The parameter MSK_DPAR_CHECK_CONVEXITY_REL_TOL can be used to relax the convexity check.

MSK_RES_ERR_CONCURRENT_OPTIMIZER

An unsupported optimizer was chosen for use with the concurrent optimizer.

MSK_RES_ERR_CONE_INDEX

An index of a non-existing cone has been specified.

MSK_RES_ERR_CONE_OVERLAP

A new cone which variables overlap with an existing cone has been specified.

MSK_RES_ERR_CONE_OVERLAP_APPEND

The cone to be appended has one variable which is already memeber of another cone.

MSK_RES_ERR_CONE_REP_VAR

A variable is included multiple times in the cone.

MSK_RES_ERR_CONE_SIZE

A cone with too few members is specified.

MSK_RES_ERR_CONE_TYPE

Invalid cone type specified.

MSK_RES_ERR_CONE_TYPE_STR

Invalid cone type specified.

MSK_RES_ERR_DATA_FILE_EXT

The data file format cannot be determined from the file name.

MSK_RES_ERR_DUP_NAME

The same name was used multiple times for the same problem item type.

MSK_RES_ERR_DUPLICATE_BARVARIABLE_NAMES

Two barvariable names are identical.

MSK_RES_ERR_DUPLICATE_CONE_NAMES

Two cone names are identical.

MSK_RES_ERR_DUPLICATE_CONSTRAINT_NAMES

Two constraint names are identical.

MSK_RES_ERR_DUPLICATE_VARIABLE_NAMES

Two variable names are identical.

MSK_RES_ERR_END_OF_FILE

End of file reached.

MSK_RES_ERR_FACTOR

An error occurred while factorizing a matrix.

MSK_RES_ERR_FEASREPAIR_CANNOT_RELAX

An optimization problem cannot be relaxed. This is the case e.g. for general nonlinear optimization problems.

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.

MSK_RES_ERR_FEASREPAIR_SOLVING_RELAXED

The relaxed problem could not be solved to optimality. Please consult the log file for further details.

MSK_RES_ERR_FILE_LICENSE

Invalid license file.

MSK_RES_ERR_FILE_OPEN

Error while opening a file.

MSK_RES_ERR_FILE_READ

File read error.

MSK_RES_ERR_FILE_WRITE

File write error.

MSK_RES_ERR_FIRST

Invalid first.

MSK_RES_ERR_FIRSTI

Invalid firsti.

MSK_RES_ERR_FIRSTJ

Invalid firstj.

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.

MSK_RES_ERR_FLEXLM

The FLEXlm license manager reported an error.

MSK_RES_ERR_GLOBAL_INV_CONIC_PROBLEM

The global optimizer can only be applied to problems without semidefinite variables.

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.

MSK_RES_ERR_HUGE_C

A huge value in absolute size is specified for one c_j.

MSK_RES_ERR_IDENTICAL_TASKS

Some tasks related to this function call were identical. Unique tasks were expected.

MSK_RES_ERR_IN_ARGUMENT

A function argument is incorrect.

MSK_RES_ERR_INDEX

An index is out of range.

MSK_RES_ERR_INDEX_ARR_IS_TOO_LARGE

An index in an array argument is too large.

MSK_RES_ERR_INDEX_ARR_IS_TOO_SMALL

An index in an array argument is too small.

MSK_RES_ERR_INDEX_IS_TOO_LARGE

An index in an argument is too large.

MSK_RES_ERR_INDEX_IS_TOO_SMALL

An index in an argument is too small.

MSK_RES_ERR_INF_DOU_INDEX

A double information index is out of range for the specified type.

MSK_RES_ERR_INF_DOU_NAME

A double information name is invalid.

MSK_RES_ERR_INF_INT_INDEX

An integer information index is out of range for the specified type.

MSK_RES_ERR_INF_INT_NAME

An integer information name is invalid.

MSK_RES_ERR_INF_LINT_INDEX

A long integer information index is out of range for the specified type.

MSK_RES_ERR_INF_LINT_NAME

A long integer information name is invalid.

MSK_RES_ERR_INF_TYPE

The information type is invalid.

MSK_RES_ERR_INFEAS_UNDEFINED

The requested value is not defined for this solution type.

MSK_RES_ERR_INFINITE_BOUND

A numerically huge bound value is specified.

MSK_RES_ERR_INT64_TO_INT32_CAST

An 32 bit integer could not cast to a 64 bit integer.

MSK_RES_ERR_INTERNAL

An internal error occurred. Please report this problem.

MSK_RES_ERR_INTERNAL_TEST_FAILED

An internal unit test function failed.

MSK_RES_ERR_INV_APTRE

`aptre[j]` is strictly smaller than `aptrb[j]` for some `j`.

MSK_RES_ERR_INV_BK

Invalid bound key.

MSK_RES_ERR_INV_BKC

Invalid bound key is specified for a constraint.

MSK_RES_ERR_INV_BKX

An invalid bound key is specified for a variable.

`MSK_RES_ERR_INV_CONE_TYPE`

Invalid cone type code is encountered.

`MSK_RES_ERR_INV_CONE_TYPE_STR`

Invalid cone type string encountered.

`MSK_RES_ERR_INV_CONIC_PROBLEM`

The conic optimizer can only be applied to problems with linear objective and constraints. Many problems such convex quadratically constrained problems can easily be reformulated to conic problems. See the appropriate MOSEK manual for details.

`MSK_RES_ERR_INV_MARKI`

Invalid value in marki.

`MSK_RES_ERR_INV_MARKJ`

Invalid value in markj.

`MSK_RES_ERR_INV_NAME_ITEM`

An invalid name item code is used.

`MSK_RES_ERR_INV_NUMI`

Invalid numi.

`MSK_RES_ERR_INV_NUMJ`

Invalid numj.

`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.

`MSK_RES_ERR_INV_PROBLEM`

Invalid problem type. Probably a nonconvex problem has been specified.

`MSK_RES_ERR_INV_QCON_SUBI`

Invalid value in `qcsubi`.

`MSK_RES_ERR_INV_QCON_SUBJ`

Invalid value in `qcsubj`.

`MSK_RES_ERR_INV_QCON_SUBK`

Invalid value in `qcsubk`.

`MSK_RES_ERR_INV_QCON_VAL`

Invalid value in `qcval`.

`MSK_RES_ERR_INV_QOBJ_SUBI`

Invalid value in `qosubi`.

MSK_RES_ERR_INV_QOBJ_SUBJ

Invalid value in qosubj.

MSK_RES_ERR_INV_QOBJ_VAL

Invalid value in qoval.

MSK_RES_ERR_INV_SK

Invalid status key code.

MSK_RES_ERR_INV_SK_STR

Invalid status key string encountered.

MSK_RES_ERR_INV_SKC

Invalid value in skc.

MSK_RES_ERR_INV_SKN

Invalid value in skn.

MSK_RES_ERR_INV_SKX

Invalid value in skx.

MSK_RES_ERR_INV_VAR_TYPE

An invalid variable type is specified for a variable.

MSK_RES_ERR_INVALID_ACCMODE

An invalid access mode is specified.

MSK_RES_ERR_INVALID_AIJ

$a_{i,j}$ contains an invalid floating point value, i.e. a NaN or an infinite value.

MSK_RES_ERR_INVALID_AMPL_STUB

Invalid AMPL stub.

MSK_RES_ERR_INVALID_BARVAR_NAME

An invalid symmetric matrix variable name is used.

MSK_RES_ERR_INVALID_BRANCH_DIRECTION

An invalid branching direction is specified.

MSK_RES_ERR_INVALID_BRANCH_PRIORITY

An invalid branching priority is specified. It should be nonnegative.

MSK_RES_ERR_INVALID_COMPRESSION

Invalid compression type.

MSK_RES_ERR_INVALID_CON_NAME

An invalid constraint name is used.

MSK_RES_ERR_INVALID_CONE_NAME

An invalid cone name is used.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_CONES

The file format does not support a problem with conic constraints.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_GENERAL_NL

The file format does not support a problem with general nonlinear terms.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_SYM_MAT

The file format does not support a problem with symmetric matrix variables.

MSK_RES_ERR_INVALID_FILE_NAME

An invalid file name has been specified.

MSK_RES_ERR_INVALID_FORMAT_TYPE

Invalid format type.

MSK_RES_ERR_INVALID_IDX

A specified index is invalid.

MSK_RES_ERR_INVALID_IOMODE

Invalid io mode.

MSK_RES_ERR_INVALID_MAX_NUM

A specified index is invalid.

MSK_RES_ERR_INVALID_NAME_IN_SOL_FILE

An invalid name occurred in a solution file.

MSK_RES_ERR_INVALID_NETWORK_PROBLEM

The problem is not a network problem as expected. The error occurs if a network optimizer is applied to a problem that cannot (easily) be converted to a network problem.

MSK_RES_ERR_INVALID_OBJ_NAME

An invalid objective name is specified.

MSK_RES_ERR_INVALID_OBJECTIVE_SENSE

An invalid objective sense is specified.

MSK_RES_ERR_INVALID_PROBLEM_TYPE

An invalid problem type.

MSK_RES_ERR_INVALID_SOL_FILE_NAME

An invalid file name has been specified.

MSK_RES_ERR_INVALID_STREAM

An invalid stream is referenced.

MSK_RES_ERR_INVALID_SURPLUS

Invalid surplus.

MSK_RES_ERR_INVALID_SYM_MAT_DIM

A sparse symmetric matrix of invalid dimension is specified.

MSK_RES_ERR_INVALID_TASK

The `task` is invalid.

MSK_RES_ERR_INVALID_UTF8

An invalid UTF8 string is encountered.

MSK_RES_ERR_INVALID_VAR_NAME

An invalid variable name is used.

MSK_RES_ERR_INVALID_WCHAR

An invalid `wchar` string is encountered.

MSK_RES_ERR_INVALID_WHICHSOL

`whichsol` is invalid.

MSK_RES_ERR_LAST

Invalid index `last`. A given index was out of expected range.

MSK_RES_ERR_LASTI

Invalid `lasti`.

MSK_RES_ERR_LASTJ

Invalid `lastj`.

MSK_RES_ERR_LAU_ARG_K

Invalid argument k.

MSK_RES_ERR_LAU_ARG_M

Invalid argument m.

MSK_RES_ERR_LAU_ARG_N

Invalid argument n.

MSK_RES_ERR_LAU_ARG_TRANS

Invalid argument trans.

MSK_RES_ERR_LAU_ARG_TRANSA

Invalid argument transa.

MSK_RES_ERR_LAU_ARG_TRANSB

Invalid argument transb.

MSK_RES_ERR_LAU_ARG_UPLO

Invalid argument uplo.

MSK_RES_ERR_LAU_SINGULAR_MATRIX

A matrix is singular.

MSK_RES_ERR_LAU_UNKNOWN

An unknown error.

MSK_RES_ERR_LICENSE

Invalid license.

MSK_RES_ERR_LICENSE_CANNOT_ALLOCATE

The license system cannot allocate the memory required.

MSK_RES_ERR_LICENSE_CANNOT_CONNECT

MOSEK cannot connect to the license server. Most likely the license server is not up and running.

MSK_RES_ERR_LICENSE_EXPIRED

The license has expired.

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.

MSK_RES_ERR_LICENSE_INVALID_HOSTID

The host ID specified in the license file does not match the host ID of the computer.

MSK_RES_ERR_LICENSE_MAX

Maximum number of licenses is reached.

MSK_RES_ERR_LICENSE_MOSEKLM_DAEMON

The MOSEKLM license manager daemon is not up and running.

MSK_RES_ERR_LICENSE_NO_SERVER_LINE

There is no `SERVER` line in the license file. All non-zero license count features need at least one `SERVER` line.

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`.

`MSK_RES_ERR_LICENSE_SERVER`

The license server is not responding.

`MSK_RES_ERR_LICENSE_SERVER_VERSION`

The version specified in the checkout request is greater than the highest version number the daemon supports.

`MSK_RES_ERR_LICENSE_VERSION`

The license is valid for another version of MOSEK.

`MSK_RES_ERR_LINK_FILE_DLL`

A file cannot be linked to a stream in the DLL version.

`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.

`MSK_RES_ERR_LOWER_BOUND_IS_A_NAN`

The lower bound specificied is not a number (nan).

`MSK_RES_ERR_LP_DUP_SLACK_NAME`

The name of the slack variable added to a ranged constraint already exists.

`MSK_RES_ERR_LP_EMPTY`

The problem cannot be written to an LP formatted file.

`MSK_RES_ERR_LP_FILE_FORMAT`

Syntax error in an LP file.

`MSK_RES_ERR_LP_FORMAT`

Syntax error in an LP file.

`MSK_RES_ERR_LP_FREE_CONSTRAINT`

Free constraints cannot be written in LP file format.

`MSK_RES_ERR_LP_INCOMPATIBLE`

The problem cannot be written to an LP formatted file.

`MSK_RES_ERR_LP_INVALID_CON_NAME`

A constraint name is invalid when used in an LP formatted file.

MSK_RES_ERR_LP_INVALID_VAR_NAME

A variable name is invalid when used in an LP formatted file.

MSK_RES_ERR_LP_WRITE_CONIC_PROBLEM

The problem contains cones that cannot be written to an LP formatted file.

MSK_RES_ERR_LP_WRITE_GECO_PROBLEM

The problem contains general convex terms that cannot be written to an LP formatted file.

MSK_RES_ERR_LU_MAX_NUM_TRIES

Could not compute the LU factors of the matrix within the maximum number of allowed tries.

MSK_RES_ERR_MAX_LEN_IS_TOO_SMALL

An maximum length that is too small has been specfified.

MSK_RES_ERR_MAXNUMBARVAR

The maximum number of semidefinite variables specified is smaller than the number of semidefinite variables in the task.

MSK_RES_ERR_MAXNUMCON

The maximum number of constraints specified is smaller than the number of constraints in the task.

MSK_RES_ERR_MAXNUMCONE

The value specified for `maxnumcone` is too small.

MSK_RES_ERR_MAXNUMQNZ

The maximum number of non-zeros specified for the Q matrixes is smaller than the number of non-zeros in the current Q matrixes.

MSK_RES_ERR_MAXNUMVAR

The maximum number of variables specified is smaller than the number of variables in the task.

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.

MSK_RES_ERR_MBT_INVALID

The MBT file is invalid.

MSK_RES_ERR_MIO_INTERNAL

A fatal error occurred in the mixed integer optimizer. Please contact MOSEK support.

MSK_RES_ERR_MIO_INVALID_NODE_OPTIMIZER

An invalid node optimizer was selected for the problem type.

MSK_RES_ERR_MIO_INVALID_ROOT_OPTIMIZER

An invalid root optimizer was selected for the problem type.

MSK_RES_ERR_MIO_NO_OPTIMIZER

No optimizer is available for the current class of integer optimization problems.

MSK_RES_ERR_MIO_NOT_LOADED

The mixed-integer optimizer is not loaded.

MSK_RES_ERR_MISSING_LICENSE_FILE

MOSEK cannot license file or a token server. See the MOSEK installation manual for details.

MSK_RES_ERR_MIXED_PROBLEM

The problem contains both conic and nonlinear constraints.

MSK_RES_ERR_MPS_CONE_OVERLAP

A variable is specified to be a member of several cones.

MSK_RES_ERR_MPS_CONE_REPEAT

A variable is repeated within the CSECTION.

MSK_RES_ERR_MPS_CONE_TYPE

Invalid cone type specified in a CSECTION.

MSK_RES_ERR_MPS_DUPLICATE_Q_ELEMENT

Duplicate elements is specfied in a Q matrix.

MSK_RES_ERR_MPS_FILE

An error occurred while reading an MPS file.

MSK_RES_ERR_MPS_INV_BOUND_KEY

An invalid bound key occurred in an MPS file.

MSK_RES_ERR_MPS_INV_CON_KEY

An invalid constraint key occurred in an MPS file.

MSK_RES_ERR_MPS_INV_FIELD

A field in the MPS file is invalid. Probably it is too wide.

MSK_RES_ERR_MPS_INV_MARKER

An invalid marker has been specified in the MPS file.

MSK_RES_ERR_MPS_INV_SEC_NAME

An invalid section name occurred in an MPS file.

MSK_RES_ERR_MPS_INV_SEC_ORDER

The sections in the MPS data file are not in the correct order.

`MSK_RES_ERR_MPS_INVALID_OBJ_NAME`

An invalid objective name is specified.

`MSK_RES_ERR_MPS_INVALID_OBJSENSE`

An invalid objective sense is specified.

`MSK_RES_ERR_MPS_MUL_CON_NAME`

A constraint name was specified multiple times in the `ROWS` section.

`MSK_RES_ERR_MPS_MUL_CSEC`

Multiple `CSECTION`s are given the same name.

`MSK_RES_ERR_MPS_MUL_QOBJ`

The Q term in the objective is specified multiple times in the MPS data file.

`MSK_RES_ERR_MPS_MUL_QSEC`

Multiple `QSECTION`s are specified for a constraint in the MPS data file.

`MSK_RES_ERR_MPS_NO_OBJECTIVE`

No objective is defined in an MPS file.

`MSK_RES_ERR_MPS_NON_SYMMETRIC_Q`

A non symmetric matrice has been speciefied.

`MSK_RES_ERR_MPS_NULL_CON_NAME`

An empty constraint name is used in an MPS file.

`MSK_RES_ERR_MPS_NULL_VAR_NAME`

An empty variable name is used in an MPS file.

`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.

`MSK_RES_ERR_MPS_TAB_IN_FIELD2`

A tab char occurred in field 2.

`MSK_RES_ERR_MPS_TAB_IN_FIELD3`

A tab char occurred in field 3.

`MSK_RES_ERR_MPS_TAB_IN_FIELD5`

A tab char occurred in field 5.

`MSK_RES_ERR_MPS_UNDEF_CON_NAME`

An undefined constraint name occurred in an MPS file.

MSK_RES_ERR_MPS_UNDEF_VAR_NAME

An undefined variable name occurred in an MPS file.

MSK_RES_ERR_MUL_A_ELEMENT

An element in A is defined multiple times.

MSK_RES_ERR_NAME_IS_NULL

The name buffer is a NULL pointer.

MSK_RES_ERR_NAME_MAX_LEN

A name is longer than the buffer that is supposed to hold it.

MSK_RES_ERR_NAN_IN_BLC

l^c contains an invalid floating point value, i.e. a `NaN`.

MSK_RES_ERR_NAN_IN_BLX

l^x contains an invalid floating point value, i.e. a `NaN`.

MSK_RES_ERR_NAN_IN_BUC

u^c contains an invalid floating point value, i.e. a `NaN`.

MSK_RES_ERR_NAN_IN_BUX

u^x contains an invalid floating point value, i.e. a `NaN`.

MSK_RES_ERR_NAN_IN_C

c contains an invalid floating point value, i.e. a `NaN`.

MSK_RES_ERR_NAN_IN_DOUBLE_DATA

An invalid floating point value was used in some double data.

MSK_RES_ERR_NEGATIVE_APPEND

Cannot append a negative number.

MSK_RES_ERR_NEGATIVE_SURPLUS

Negative surplus.

MSK_RES_ERR_NEWER_DLL

The dynamic link library is newer than the specified version.

MSK_RES_ERR_NO_BARS_FOR_SOLUTION

There is no $\bar{s}$ available for the solution specified. In particular note there are no $\bar{s}$ defined for the basic and integer solutions.

MSK_RES_ERR_NO_BARX_FOR_SOLUTION

There is no $\bar{X}$ available for the solution specified. In particular note there are no $\bar{X}$ defined for the basic and integer solutions.

MSK_RES_ERR_NO_BASIS_SOL

No basic solution is defined.

MSK_RES_ERR_NO_DUAL_FOR_ITG_SOL

No dual information is available for the integer solution.

MSK_RES_ERR_NO_DUAL_INFEAS_CER

A certificate of infeasibility is not available.

MSK_RES_ERR_NO_DUAL_INFO_FOR_ITG_SOL

Dual information is not available for the integer solution.

MSK_RES_ERR_NO_INIT_ENV

`env` is not initialized.

MSK_RES_ERR_NO_OPTIMIZER_VAR_TYPE

No optimizer is available for this class of optimization problems.

MSK_RES_ERR_NO_PRIMAL_INFEAS_CER

A certificate of primal infeasibility is not available.

MSK_RES_ERR_NO_SNX_FOR_BAS_SOL

s_n^x is not available for the basis solution.

MSK_RES_ERR_NO_SOLUTION_IN_CALLBACK

The required solution is not available.

MSK_RES_ERR_NON_UNIQUE_ARRAY

An array does not contain unique elements.

MSK_RES_ERR_NONCONVEX

The optimization problem is nonconvex.

MSK_RES_ERR_NONLINEAR_EQUALITY

The model contains a nonlinear equality which defines a nonconvex set.

MSK_RES_ERR_NONLINEAR_FUNCTIONS_NOT_ALLOWED

An operation that is invalid for problems with nonlinear functions defined has been attempted.

MSK_RES_ERR_NONLINEAR_RANGED

The model contains a nonlinear ranged constraint which by definition defines a nonconvex set.

MSK_RES_ERR_NR_ARGUMENTS

Incorrect number of function arguments.

MSK_RES_ERR_NULL_ENV

`env` is a NULL pointer.

MSK_RES_ERR_NULL_POINTER

An argument to a function is unexpectedly a NULL pointer.

MSK_RES_ERR_NULL_TASK

`task` is a NULL pointer.

MSK_RES_ERR_NUMCONLIM

Maximum number of constraints limit is exceeded.

MSK_RES_ERR_NUMVARLIM

Maximum number of variables limit is exceeded.

MSK_RES_ERR_OBJ_Q_NOT_NSD

The quadratic coefficient matrix in the objective is not negative semidefinite as expected for a maximization problem. The parameter MSK_DPAR_CHECK_CONVEXITY_REL_TOL can be used to relax the convexity check.

MSK_RES_ERR_OBJ_Q_NOT_PSD

The quadratic coefficient matrix in the objective is not positive semidefinite as expected for a minimization problem. The parameter MSK_DPAR_CHECK_CONVEXITY_REL_TOL can be used to relax the convexity check.

MSK_RES_ERR_OBJECTIVE_RANGE

Empty objective range.

MSK_RES_ERR_OLDER_DLL

The dynamic link library is older than the specified version.

MSK_RES_ERR_OPEN_DL

A dynamic link library could not be opened.

MSK_RES_ERR_OPF_FORMAT

Syntax error in an OPF file

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.

MSK_RES_ERR_OPF_PREMATURE_EOF

Premature end of file in an OPF file.

MSK_RES_ERR_OPTIMIZER_LICENSE

The optimizer required is not licensed.

MSK_RES_ERR_ORD_INVALID

Invalid content in branch ordering file.

MSK_RES_ERR_ORD_INVALID_BRANCH_DIR

An invalid branch direction key is specified.

MSK_RES_ERR_OVERFLOW

A computation produced an overflow i.e. a very large number.

MSK_RES_ERR_PARAM_INDEX

Parameter index is out of range.

MSK_RES_ERR_PARAM_IS_TOO_LARGE

The parameter value is too large.

MSK_RES_ERR_PARAM_IS_TOO_SMALL

The parameter value is too small.

MSK_RES_ERR_PARAM_NAME

The parameter name is not correct.

MSK_RES_ERR_PARAM_NAME_DOU

The parameter name is not correct for a double parameter.

MSK_RES_ERR_PARAM_NAME_INT

The parameter name is not correct for an integer parameter.

MSK_RES_ERR_PARAM_NAME_STR

The parameter name is not correct for a string parameter.

MSK_RES_ERR_PARAM_TYPE

The parameter type is invalid.

MSK_RES_ERR_PARAM_VALUE_STR

The parameter value string is incorrect.

MSK_RES_ERR_PLATFORM_NOT_LICENSED

A requested license feature is not available for the required platform.

MSK_RES_ERR_POSTSOLVE

An error occurred during the postsolve. Please contact MOSEK support.

MSK_RES_ERR_PRO_ITEM

An invalid problem is used.

MSK_RES_ERR_PROB_LICENSE

The software is not licensed to solve the problem.

MSK_RES_ERR_QCON_SUBI_TOO_LARGE

Invalid value in `qcsubi`.

MSK_RES_ERR_QCON_SUBI_TOO_SMALL

Invalid value in qcsubi.

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.

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.

MSK_RES_ERR_READ_FORMAT

The specified format cannot be read.

MSK_RES_ERR_READ_LP_MISSING_END_TAG

Syntax error in LP file. Possibly missing End tag.

MSK_RES_ERR_READ_LP_NONEXISTING_NAME

A variable never occurred in objective or constraints.

MSK_RES_ERR_REMOVE_CONE_VARIABLE

A variable cannot be removed because it will make a cone invalid.

MSK_RES_ERR_REPAIR_INVALID_PROBLEM

The feasibility repair does not support the specified problem type.

MSK_RES_ERR_REPAIR_OPTIMIZATION_FAILED

Computation the optimal relaxation failed. The cause may have been numerical problems.

MSK_RES_ERR_SEN_BOUND_INVALID_LO

Analysis of lower bound requested for an index, where no lower bound exists.

MSK_RES_ERR_SEN_BOUND_INVALID_UP

Analysis of upper bound requested for an index, where no upper bound exists.

MSK_RES_ERR_SEN_FORMAT

Syntax error in sensitivity analysis file.

MSK_RES_ERR_SEN_INDEX_INVALID

Invalid range given in the sensitivity file.

MSK_RES_ERR_SEN_INDEX_RANGE

Index out of range in the sensitivity analysis file.

MSK_RES_ERR_SEN_INVALID_REGEXP

Syntax error in regexp or regexp longer than 1024.

MSK_RES_ERR_SEN_NUMERICAL

Numerical difficulties encountered performing the sensitivity analysis.

MSK_RES_ERR_SEN_SOLUTION_STATUS

No optimal solution found to the original problem given for sensitivity analysis.

MSK_RES_ERR_SEN_UNDEF_NAME

An undefined name was encountered in the sensitivity analysis file.

MSK_RES_ERR_SEN_UNHANDLED_PROBLEM_TYPE

Sensitivity analysis cannot be performed for the spcified problem. Sensitivity analysis is only possible for linear problems.

MSK_RES_ERR_SIZE_LICENSE

The problem is bigger than the license.

MSK_RES_ERR_SIZE_LICENSE_CON

The problem has too many constraints to be solved with the available license.

MSK_RES_ERR_SIZE_LICENSE_INTVAR

The problem contains too many integer variables to be solved with the available license.

MSK_RES_ERR_SIZE_LICENSE_NUMCORES

The computer contains more cpu cores than the license allows for.

MSK_RES_ERR_SIZE_LICENSE_VAR

The problem has too many variables to be solved with the available license.

MSK_RES_ERR_SOL_FILE_INVALID_NUMBER

An invalid number is specified in a solution file.

MSK_RES_ERR_SOLITEM

The solution item number `solitem` is invalid. Please note that MSK_SOL_ITEM_SNX is invalid for the basic solution.

MSK_RES_ERR_SOLVER_PROBTYPE

Problem type does not match the chosen optimizer.

MSK_RES_ERR_SPACE

Out of space.

MSK_RES_ERR_SPACE_LEAKING

MOSEK is leaking memory. This can be due to either an incorrect use of MOSEK or a bug.

MSK_RES_ERR_SPACE_NO_INFO

No available information about the space usage.

MSK_RES_ERR_SYM_MAT_DUPLICATE

A value in a symmetric matric as been specified more than once.

MSK_RES_ERR_SYM_MAT_INVALID_COL_INDEX

A column index specified for sparse symmetric maxtrix is invalid.

MSK_RES_ERR_SYM_MAT_INVALID_ROW_INDEX

A row index specified for sparse symmetric maxtrix is invalid.

MSK_RES_ERR_SYM_MAT_INVALID_VALUE

The numerical value specified in a sparse symmetric matrix is not a value floating value.

MSK_RES_ERR_SYM_MAT_NOT_LOWER_TRINGULAR

Only the lower triangular part of sparse symmetric matrix should be specified.

MSK_RES_ERR_TASK_INCOMPATIBLE

The Task file is incompatible with this platform. This results from reading a file on a 32 bit platform generated on a 64 bit platform.

MSK_RES_ERR_TASK_INVALID

The Task file is invalid.

MSK_RES_ERR_THREAD_COND_INIT

Could not initialize a condition.

MSK_RES_ERR_THREAD_CREATE

Could not create a thread. This error may occur if a large number of environments are created and not deleted again. In any case it is a good practice to minimize the number of environments created.

MSK_RES_ERR_THREAD_MUTEX_INIT

Could not initialize a mutex.

MSK_RES_ERR_THREAD_MUTEX_LOCK

Could not lock a mutex.

MSK_RES_ERR_THREAD_MUTEX_UNLOCK

Could not unlock a mutex.

MSK_RES_ERR_TOCONIC_CONVERSION_FAIL

A constraint could not be converted in conic form.

MSK_RES_ERR_TOO_MANY_CONCURRENT_TASKS

Too many concurrent tasks specified.

MSK_RES_ERR_TOO_SMALL_MAX_NUM_NZ

The maximum number of non-zeros specified is too small.

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.

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.

MSK_RES_ERR_UNDEF_SOLUTION

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.

MSK_RES_ERR_UNDEFINED_OBJECTIVE_SENSE

The objective sense has not been specified before the optimization.

MSK_RES_ERR_UNHANDLED_SOLUTION_STATUS

Unhandled solution status.

MSK_RES_ERR_UNKNOWN

Unknown error.

MSK_RES_ERR_UPPER_BOUND_IS_A_NAN

The upper bound specificied is not a number (nan).

MSK_RES_ERR_UPPER_TRIANGLE

An element in the upper triangle of a lower triangular matrix is specified.

MSK_RES_ERR_USER_FUNC_RET

An user function reported an error.

MSK_RES_ERR_USER_FUNC_RET_DATA

An user function returned invalid data.

MSK_RES_ERR_USER_NLO_EVAL

The user-defined nonlinear function reported an error.

MSK_RES_ERR_USER_NLO_EVAL_HESSUBI

The user-defined nonlinear function reported an invalid subscript in the Hessian.

MSK_RES_ERR_USER_NLO_EVAL_HESSUBJ

The user-defined nonlinear function reported an invalid subscript in the Hessian.

MSK_RES_ERR_USER_NLO_FUNC

The user-defined nonlinear function reported an error.

MSK_RES_ERR_WHICHITEM_NOT_ALLOWED

`whichitem` is unacceptable.

MSK_RES_ERR_WHICHSOL

The solution defined by compwhichsol does not exists.

MSK_RES_ERR_WRITE_LP_FORMAT

Problem cannot be written as an LP file.

MSK_RES_ERR_WRITE_LP_NON_UNIQUE_NAME

An auto-generated name is not unique.

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.

MSK_RES_ERR_WRITE_OPF_INVALID_VAR_NAME

Empty variable names cannot be written to OPF files.

MSK_RES_ERR_WRITING_FILE

An error occurred while writing file

MSK_RES_ERR_XML_INVALID_PROBLEM_TYPE

The problem type is not supported by the XML format.

MSK_RES_ERR_Y_IS_UNDEFINED

The solution item y is undefined.

MSK_RES_OK

No error occurred.

MSK_RES_TRM_INTERNAL

The optimizer terminated due to some internal reason. Please contact MOSEK support.

MSK_RES_TRM_INTERNAL_STOP

The optimizer terminated for internal reasons. Please contact MOSEK support.

MSK_RES_TRM_MAX_ITERATIONS

The optimizer terminated at the maximum number of iterations.

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.

MSK_RES_TRM_MAX_TIME

The optimizer terminated at the maximum amount of time.

MSK_RES_TRM_MIO_NEAR_ABS_GAP

The mixed-integer optimizer terminated because the near optimal absolute gap tolerance was satisfied.

MSK_RES_TRM_MIO_NEAR_REL_GAP

The mixed-integer optimizer terminated because the near optimal relative gap tolerance was satisfied.

MSK_RES_TRM_MIO_NUM_BRANCHES

The mixed-integer optimizer terminated as to the maximum number of branches was reached.

MSK_RES_TRM_MIO_NUM_RELAXS

The mixed-integer optimizer terminated as the maximum number of relaxations was reached.

MSK_RES_TRM_NUM_MAX_NUM_INT_SOLUTIONS

The mixed-integer optimizer terminated as the maximum number of feasible solutions was reached.

MSK_RES_TRM_NUMERICAL_PROBLEM

The optimizer terminated due to numerical problems.

MSK_RES_TRM_OBJECTIVE_RANGE

The optimizer terminated on the bound of the objective range.

MSK_RES_TRM_STALL

The optimizer is terminated due to slow progress.

Stalling means that numerical problems prevent the optimizer from making reasonable progress and that it make no sense to continue. In many cases this happens if the problem is badly scaled or otherwise ill-conditioned. There is no guarantee that the solution will be (near) feasible or near optimal. However, often stalling happens near the optimum, and the returned solution may be of good quality. Therefore, it is recommended to check the status of then solution. If the solution near optimal the solution is most likely good enough for most practical purposes.

Please note that if a linear optimization problem is solved using the interior-point optimizer with basis identification turned on, the returned basic solution likely to have high accuracy, even though the optimizer stalled.

Some common causes of stalling are a) badly scaled models, b) near feasible or near infeasible problems and c) a non-convex problems. Case c) is only relevant for general non-linear problems. It is not possible in general for MOSEK to check if a specific problems is convex since such a check would be NP hard in itself. This implies that care should be taken when solving problems involving general user defined functions.

MSK_RES_TRM_USER_CALLBACK

The optimizer terminated due to the return of the user-defined call-back function.

MSK_RES_WRN_ANA_ALMOST_INT_BOUNDS

This warning is issued by the problem analyzer if a constraint is bound nearly integral.

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.

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.

MSK_RES_WRN_ANA_EMPTY_COLS

This warning is issued by the problem analyzer, if columns, in which all coefficients are zero, are found.

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.

MSK_RES_WRN_CONSTRUCT_INVALID_SOL_ITG

The intial value for one or more of the integer variables is not feasible.

MSK_RES_WRN_CONSTRUCT_NO_SOL_ITG

The construct solution requires an integer solution.

MSK_RES_WRN_CONSTRUCT_SOLUTION_INFEAS

After fixing the integer variables at the suggested values then the problem is infeasible.

MSK_RES_WRN_DROPPED_NZ_QOBJ

One or more non-zero elements were dropped in the Q matrix in the objective.

MSK_RES_WRN_DUPLICATE_BARVARIABLE_NAMES

Two barvariable names are identical.

MSK_RES_WRN_DUPLICATE_CONE_NAMES

Two cone names are identical.

MSK_RES_WRN_DUPLICATE_CONSTRAINT_NAMES

Two constraint names are identical.

MSK_RES_WRN_DUPLICATE_VARIABLE_NAMES

Two variable names are identical.

MSK_RES_WRN_ELIMINATOR_SPACE

The eliminator is skipped at least once due to lack of space.

MSK_RES_WRN_EMPTY_NAME

A variable or constraint name is empty. The output file may be invalid.

MSK_RES_WRN_IGNORE_INTEGER

Ignored integer constraints.

MSK_RES_WRN_INCOMPLETE_LINEAR_DEPENDENCY_CHECK

The linear dependency check(s) is not completed. Normally this is not an important warning unless the optimization problem has been formulated with linear dependencies which is bad practice.

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.

MSK_RES_WRN_LARGE_BOUND

A numerically large bound value is specified.

MSK_RES_WRN_LARGE_CJ

A numerically large value is specified for one c_j.

MSK_RES_WRN_LARGE_CON_FX

An equality constraint is fixed to a numerically large value. This can cause numerical problems.

MSK_RES_WRN_LARGE_LO_BOUND

A numerically large lower bound value is specified.

MSK_RES_WRN_LARGE_UP_BOUND

A numerically large upper bound value is specified.

MSK_RES_WRN_LICENSE_EXPIRE

The license expires.

MSK_RES_WRN_LICENSE_FEATURE_EXPIRE

The license expires.

MSK_RES_WRN_LICENSE_SERVER

The license server is not responding.

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.

MSK_RES_WRN_LP_OLD_QUAD_FORMAT

Missing '/2' after quadratic expressions in bound or objective.

MSK_RES_WRN_MIO_INFEASIBLE_FINAL

The final mixed-integer problem with all the integer variables fixed at their optimal values is infeasible.

MSK_RES_WRN_MPS_SPLIT_BOU_VECTOR

A BOUNDS vector is split into several nonadjacent parts in an MPS file.

MSK_RES_WRN_MPS_SPLIT_RAN_VECTOR

A RANGE vector is split into several nonadjacent parts in an MPS file.

MSK_RES_WRN_MPS_SPLIT_RHS_VECTOR

An RHS vector is split into several nonadjacent parts in an MPS file.

MSK_RES_WRN_NAME_MAX_LEN

A name is longer than the buffer that is supposed to hold it.

MSK_RES_WRN_NO_DUALIZER

No automatic dualizer is available for the specified problem. The primal problem is solved.

MSK_RES_WRN_NO_GLOBAL_OPTIMIZER

No global optimizer is available.

MSK_RES_WRN_NO_NONLINEAR_FUNCTION_WRITE

The problem contains a general nonlinear function in either the objective or the constraints. Such a nonlinear function cannot be written to a disk file. Note that quadratic terms when inputted explicitly can be written to disk.

MSK_RES_WRN_NZ_IN_UPR_TRI

Non-zero elements specified in the upper triangle of a matrix were ignored.

MSK_RES_WRN_OPEN_PARAM_FILE

The parameter file could not be opened.

MSK_RES_WRN_PARAM_IGNORED_CMIO

A parameter was ignored by the conic mixed integer optimizer.

MSK_RES_WRN_PARAM_NAME_DOU

The parameter name is not recognized as a double parameter.

MSK_RES_WRN_PARAM_NAME_INT

The parameter name is not recognized as a integer parameter.

MSK_RES_WRN_PARAM_NAME_STR

The parameter name is not recognized as a string parameter.

MSK_RES_WRN_PARAM_STR_VALUE

The string is not recognized as a symbolic value for the parameter.

MSK_RES_WRN_PRESOLVE_OUTOFSPACE

The presolve is incomplete due to lack of space.

MSK_RES_WRN_QUAD_CONES_WITH_ROOT_FIXED_AT_ZERO

For at least one quadratic cone the root is fixed at (nearly) zero. This may cause problems such as a very large dual solution. Therefore, it is recommended to remove such cones before optimizing the problems, or to fix all the variables in the cone to 0.

MSK_RES_WRN_RQUAD_CONES_WITH_ROOT_FIXED_AT_ZERO

For at least one rotated quadratic cone at least one of the root variables are fixed at (nearly) zero. This may cause problems such as a very large dual solution. Therefore, it is recommended to remove such cones before optimizing the problems, or to fix all the variables in the cone to 0.

MSK_RES_WRN_SOL_FILE_IGNORED_CON

One or more lines in the constraint section were ignored when reading a solution file.

MSK_RES_WRN_SOL_FILE_IGNORED_VAR

One or more lines in the variable section were ignored when reading a solution file.

MSK_RES_WRN_SOL_FILTER

Invalid solution filter is specified.

MSK_RES_WRN_SPAR_MAX_LEN

A value for a string parameter is longer than the buffer that is supposed to hold it.

MSK_RES_WRN_TOO_FEW_BASIS_VARS

An incomplete basis has been specified. Too few basis variables are specified.

MSK_RES_WRN_TOO_MANY_BASIS_VARS

A basis with too many variables has been specified.

MSK_RES_WRN_TOO_MANY_THREADS_CONCURRENT

The concurrent optimizer employs more threads than available. This will lead to poor performance.

MSK_RES_WRN_UNDEF_SOL_FILE_NAME

Undefined name occurred in a solution.

MSK_RES_WRN_USING_GENERIC_NAMES

Generic names are used because a name is not valid. For instance when writing an LP file the names must not contain blanks or start with a digit.

MSK_RES_WRN_WRITE_CHANGED_NAMES

Some names were changed because they were invalid for the output file format.

MSK_RES_WRN_WRITE_DISCARDED_CFIX

The fixed objective term could not be converted to a variable and was discarded in the output file.

MSK_RES_WRN_ZERO_AIJ

One or more zero elements are specified in A.

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.

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.

Chapter 11

API constants

11.1 Constraint or variable access modes

MSK_ACC_VAR

Access data by columns (variable oriented)

MSK_ACC_CON

Access data by rows (constraint oriented)

11.2 Basis identification

MSK_BI_NEVER

Never do basis identification.

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_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_RESERVERED

Not currently in use.

11.3 Bound keys

MSK_BK_LO

The constraint or variable has a finite lower bound and an infinite upper bound.

MSK_BK_UP

The constraint or variable has an infinite lower bound and an finite upper bound.

MSK_BK_FX

The constraint or variable is fixed.

MSK_BK_FR

The constraint or variable is free.

MSK_BK_RA

The constraint or variable is ranged.

11.4 Specifies the branching direction.

MSK_BRANCH_DIR_FREE

The mixed-integer optimizer decides which branch to choose.

MSK_BRANCH_DIR_UP

The mixed-integer optimizer always chooses the up branch first.

MSK_BRANCH_DIR_DOWN

The mixed-integer optimizer always chooses the down branch first.

11.5 Progress call-back codes

MSK_CALLBACK_BEGIN_BI

The basis identification procedure has been started.

MSK_CALLBACK_BEGIN_CONCURRENT

Concurrent optimizer is started.

MSK_CALLBACK_BEGIN_CONIC

The call-back function is called when the conic optimizer is started.

MSK_CALLBACK_BEGIN_DUAL_BI

The call-back function is called from within the basis identification procedure when the dual phase is started.

MSK_CALLBACK_BEGIN_DUAL_SENSITIVITY

Dual sensitivity analysis is started.

MSK_CALLBACK_BEGIN_DUAL_SETUP_BI

The call-back function is called when the dual BI phase is started.

MSK_CALLBACK_BEGIN_DUAL_SIMPLEX

The call-back function is called when the dual simplex optimizer started.

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.

MSK_CALLBACK_BEGIN_FULL_CONVEXITY_CHECK

Begin full convexity check.

MSK_CALLBACK_BEGIN_INFEAS_ANA

The call-back function is called when the infeasibility analyzer is started.

MSK_CALLBACK_BEGIN_INTPNT

The call-back function is called when the interior-point optimizer is started.

MSK_CALLBACK_BEGIN_LICENSE_WAIT

Begin waiting for license.

MSK_CALLBACK_BEGIN_MIO

The call-back function is called when the mixed-integer optimizer is started.

MSK_CALLBACK_BEGIN_NETWORK_DUAL_SIMPLEX

The call-back function is called when the dual network simplex optimizer is started.

MSK_CALLBACK_BEGIN_NETWORK_PRIMAL_SIMPLEX

The call-back function is called when the primal network simplex optimizer is started.

MSK_CALLBACK_BEGIN_NETWORK_SIMPLEX

The call-back function is called when the simplex network optimizer is started.

MSK_CALLBACK_BEGIN_NONCONVEX

The call-back function is called when the nonconvex optimizer is started.

MSK_CALLBACK_BEGIN_OPTIMIZER

The call-back function is called when the optimizer is started.

MSK_CALLBACK_BEGIN_PRESOLVE

The call-back function is called when the presolve is started.

MSK_CALLBACK_BEGIN_PRIMAL_BI

The call-back function is called from within the basis identification procedure when the primal phase is started.

MSK_CALLBACK_BEGIN_PRIMAL_DUAL_SIMPLEX

The call-back function is called when the primal-dual simplex optimizer is started.

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.

MSK_CALLBACK_BEGIN_PRIMAL_REPAIR

Begin primal feasibility repair.

MSK_CALLBACK_BEGIN_PRIMAL_SENSITIVITY

Primal sensitivity analysis is started.

MSK_CALLBACK_BEGIN_PRIMAL_SETUP_BI

The call-back function is called when the primal BI setup is started.

MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX

The call-back function is called when the primal simplex optimizer is started.

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.

MSK_CALLBACK_BEGIN_QCQO_REFORMULATE

Begin QCQO reformulation.

MSK_CALLBACK_BEGIN_READ

MOSEK has started reading a problem file.

MSK_CALLBACK_BEGIN_SIMPLEX

The call-back function is called when the simplex optimizer is started.

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.

MSK_CALLBACK_BEGIN_SIMPLEX_NETWORK_DETECT

The call-back function is called when the network detection procedure is started.

MSK_CALLBACK_BEGIN_WRITE

MOSEK has started writing a problem file.

`MSK_CALLBACK_CONIC`

The call-back function is called from within the conic optimizer after the information database has been updated.

`MSK_CALLBACK_DUAL_SIMPLEX`

The call-back function is called from within the dual simplex optimizer.

`MSK_CALLBACK_END_BI`

The call-back function is called when the basis identification procedure is terminated.

`MSK_CALLBACK_END_CONCURRENT`

Concurrent optimizer is terminated.

`MSK_CALLBACK_END_CONIC`

The call-back function is called when the conic optimizer is terminated.

`MSK_CALLBACK_END_DUAL_BI`

The call-back function is called from within the basis identification procedure when the dual phase is terminated.

`MSK_CALLBACK_END_DUAL_SENSITIVITY`

Dual sensitivity analysis is terminated.

`MSK_CALLBACK_END_DUAL_SETUP_BI`

The call-back function is called when the dual BI phase is terminated.

`MSK_CALLBACK_END_DUAL_SIMPLEX`

The call-back function is called when the dual simplex optimizer is terminated.

`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.

`MSK_CALLBACK_END_FULL_CONVEXITY_CHECK`

End full convexity check.

`MSK_CALLBACK_END_INFEAS_ANA`

The call-back function is called when the infeasibility analyzer is terminated.

`MSK_CALLBACK_END_INTPNT`

The call-back function is called when the interior-point optimizer is terminated.

`MSK_CALLBACK_END_LICENSE_WAIT`

End waiting for license.

`MSK_CALLBACK_END_MIO`

The call-back function is called when the mixed-integer optimizer is terminated.

MSK_CALLBACK_END_NETWORK_DUAL_SIMPLEX

The call-back function is called when the dual network simplex optimizer is terminated.

MSK_CALLBACK_END_NETWORK_PRIMAL_SIMPLEX

The call-back function is called when the primal network simplex optimizer is terminated.

MSK_CALLBACK_END_NETWORK_SIMPLEX

The call-back function is called when the simplex network optimizer is terminated.

MSK_CALLBACK_END_NONCONVEX

The call-back function is called when the nonconvex optimizer is terminated.

MSK_CALLBACK_END_OPTIMIZER

The call-back function is called when the optimizer is terminated.

MSK_CALLBACK_END_PRESOLVE

The call-back function is called when the presolve is completed.

MSK_CALLBACK_END_PRIMAL_BI

The call-back function is called from within the basis identification procedure when the primal phase is terminated.

MSK_CALLBACK_END_PRIMAL_DUAL_SIMPLEX

The call-back function is called when the primal-dual simplex optimizer is terminated.

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.

MSK_CALLBACK_END_PRIMAL_REPAIR

End primal feasibility repair.

MSK_CALLBACK_END_PRIMAL_SENSITIVITY

Primal sensitivity analysis is terminated.

MSK_CALLBACK_END_PRIMAL_SETUP_BI

The call-back function is called when the primal BI setup is terminated.

MSK_CALLBACK_END_PRIMAL_SIMPLEX

The call-back function is called when the primal simplex optimizer is terminated.

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.

MSK_CALLBACK_END_QCQO_REFORMULATE

End QCQO reformulation.

MSK_CALLBACK_END_READ

MOSEK has finished reading a problem file.

MSK_CALLBACK_END_SIMPLEX

The call-back function is called when the simplex optimizer is terminated.

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.

MSK_CALLBACK_END_SIMPLEX_NETWORK_DETECT

The call-back function is called when the network detection procedure is terminated.

MSK_CALLBACK_END_WRITE

MOSEK has finished writing a problem file.

MSK_CALLBACK_IM_BI

The call-back function is called from within the basis identification procedure at an intermediate point.

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.

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.

MSK_CALLBACK_IM_DUAL_SENSIVITY

The call-back function is called at an intermediate stage of the dual sensitivity analysis.

MSK_CALLBACK_IM_DUAL_SIMPLEX

The call-back function is called at an intermediate point in the dual simplex optimizer.

MSK_CALLBACK_IM_FULL_CONVEXITY_CHECK

The call-back function is called at an intermediate stage of the full convexity check.

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.

MSK_CALLBACK_IM_LICENSE_WAIT

MOSEK is waiting for a license.

MSK_CALLBACK_IM_LU

The call-back function is called from within the LU factorization procedure at an intermediate point.

MSK_CALLBACK_IM_MIO

The call-back function is called at an intermediate point in the mixed-integer optimizer.

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.

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.

MSK_CALLBACK_IM_MIO_PRESOLVE

The call-back function is called at an intermediate point in the mixed-integer optimizer while running the presolve.

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.

MSK_CALLBACK_IM_NETWORK_DUAL_SIMPLEX

The call-back function is called at an intermediate point in the dual network simplex optimizer.

MSK_CALLBACK_IM_NETWORK_PRIMAL_SIMPLEX

The call-back function is called at an intermediate point in the primal network simplex optimizer.

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.

MSK_CALLBACK_IM_ORDER

The call-back function is called from within the matrix ordering procedure at an intermediate point.

MSK_CALLBACK_IM_PRESOLVE

The call-back function is called from within the presolve procedure at an intermediate stage.

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.

MSK_CALLBACK_IM_PRIMAL_DUAL_SIMPLEX

The call-back function is called at an intermediate point in the primal-dual simplex optimizer.

MSK_CALLBACK_IM_PRIMAL_SENSIVITY

The call-back function is called at an intermediate stage of the primal sensitivity analysis.

MSK_CALLBACK_IM_PRIMAL_SIMPLEX

The call-back function is called at an intermediate point in the primal simplex optimizer.

MSK_CALLBACK_IM_QO_REFORMULATE

The call-back function is called at an intermediate stage of the conic quadratic reformulation.

MSK_CALLBACK_IM_READ

Intermediate stage in reading.

MSK_CALLBACK_IM_SIMPLEX

The call-back function is called from within the simplex optimizer at an intermediate point.

MSK_CALLBACK_IM_SIMPLEX_BI

The call-back function is called from within the basis identification procedure at an intermediate point in the simplex clean-up phase. The frequency of the call-backs is controlled by the MSK_IPAR_LOG_SIM_FREQ parameter.

MSK_CALLBACK_INTPNT

The call-back function is called from within the interior-point optimizer after the information database has been updated.

MSK_CALLBACK_NEW_INT_MIO

The call-back function is called after a new integer solution has been located by the mixed-integer optimizer.

MSK_CALLBACK_NONCOVEX

The call-back function is called from within the nonconvex optimizer after the information database has been updated.

MSK_CALLBACK_PRIMAL_SIMPLEX

The call-back function is called from within the primal simplex optimizer.

MSK_CALLBACK_READ_OPF

The call-back function is called from the OPF reader.

MSK_CALLBACK_READ_OPF_SECTION

A chunk of Q non-zeos has been read from a problem file.

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.

MSK_CALLBACK_UPDATE_DUAL_SIMPLEX

The call-back function is called in the dual simplex optimizer.

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.

MSK_CALLBACK_UPDATE_NETWORK_DUAL_SIMPLEX

The call-back function is called in the dual network simplex optimizer.

MSK_CALLBACK_UPDATE_NETWORK_PRIMAL_SIMPLEX

The call-back function is called in the primal network simplex optimizer.

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.

MSK_CALLBACK_UPDATE_PRESOLVE

The call-back function is called from within the presolve procedure.

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.

MSK_CALLBACK_UPDATE_PRIMAL_DUAL_SIMPLEX

The call-back function is called in the primal-dual simplex optimizer.

MSK_CALLBACK_UPDATE_PRIMAL_DUAL_SIMPLEX_BI

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.

MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX

The call-back function is called in the primal simplex optimizer.

MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX_BI

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.

MSK_CALLBACK_WRITE_OPF

The call-back function is called from the OPF writer.

11.6 Types of convexity checks.

`MSK_CHECK_CONVEXITY_NONE`

No convexity check.

`MSK_CHECK_CONVEXITY_SIMPLE`

Perform simple and fast convexity check.

`MSK_CHECK_CONVEXITY_FULL`

Perform a full convexity check.

11.7 Compression types

`MSK_COMPRESS_NONE`

No compression is used.

`MSK_COMPRESS_FREE`

The type of compression used is chosen automatically.

`MSK_COMPRESS_GZIP`

The type of compression used is gzip compatible.

11.8 Cone types

`MSK_CT_QUAD`

The cone is a quadratic cone.

`MSK_CT_RQUAD`

The cone is a rotated quadratic cone.

11.9 Data format types

`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_OP

The data file is an optimization problem formatted file.

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_TASK

Generic task dump file.

MSK_DATA_FORMAT_CB

Conic benchmark format.

11.10 Double information items

MSK_DINF_BI_CLEAN_DUAL_TIME

Time spent within the dual clean-up optimizer of the basis identification procedure since its invocation.

MSK_DINF_BI_CLEAN_PRIMAL_DUAL_TIME

Time spent within the primal-dual clean-up optimizer of the basis identification procedure since its invocation.

MSK_DINF_BI_CLEAN_PRIMAL_TIME

Time spent within the primal clean-up optimizer of the basis identification procedure since its invocation.

MSK_DINF_BI_CLEAN_TIME

Time spent within the clean-up phase of the basis identification procedure since its invocation.

MSK_DINF_BI_DUAL_TIME

Time spent within the dual phase basis identification procedure since its invocation.

MSK_DINF_BI_PRIMAL_TIME

Time spent within the primal phase of the basis identification procedure since its invocation.

MSK_DINF_BI_TIME

Time spent within the basis identification procedure since its invocation.

MSK_DINF_CONCURRENT_TIME

Time spent within the concurrent optimizer since its invocation.

MSK_DINF_INTPNT_DUAL_FEAS

Dual feasibility measure reported by the interior-point optimizer. (For the interior-point optimizer this measure does not directly related to the original problem because a homogeneous model is employed.)

MSK_DINF_INTPNT_DUAL_OBJ

Dual objective value reported by the interior-point optimizer.

MSK_DINF_INTPNT_FACTOR_NUM_FLOPS

An estimate of the number of flops used in the factorization.

MSK_DINF_INTPNT_OPT_STATUS

This measure should converge to +1 if the problem has a primal-dual optimal solution, and converge to -1 if problem is (strictly) primal or dual infeasible. Furthermore, if the measure converges to 0 the problem is usually ill-posed.

MSK_DINF_INTPNT_ORDER_TIME

Order time (in seconds).

MSK_DINF_INTPNT_PRIMAL_FEAS

Primal feasibility measure reported by the interior-point optimizers. (For the interior-point optimizer this measure does not directly related to the original problem because a homogeneous model is employed).

MSK_DINF_INTPNT_PRIMAL_OBJ

Primal objective value reported by the interior-point optimizer.

MSK_DINF_INTPNT_TIME

Time spent within the interior-point optimizer since its invocation.

MSK_DINF_MIO_CG_SEPERATION_TIME

Seperation time for CG cuts.

MSK_DINF_MIO_CMIR_SEPERATION_TIME

Seperation time for CMIR cuts.

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.

MSK_DINF_MIO_DUAL_BOUND_AFTER_PRESOLVE

Value of the dual bound after presolve but before cut generation.

MSK_DINF_MIO_HEURISTIC_TIME

Time spent in the optimizer while solving the relaxtions.

MSK_DINF_MIO_OBJ_ABS_GAP

Given the mixed-integer optimizer has computed a feasible solution and a bound on the optimal objective value, then this item contains the absolute gap defined by

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

Otherwise it has the value -1.0.

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.

MSK_DINF_MIO_OBJ_INT

The primal objective value corresponding to the best integer feasible solution. Please note that at least one integer feasible solution must have located i.e. check MSK_IINF_MIO_NUM_INT_SOLUTIONS.

MSK_DINF_MIO_OBJ_REL_GAP

Given that the mixed-integer optimizer has computed a feasible solution and a bound on the optimal objective value, then this item contains the relative gap defined by

$$\frac{|(\text{objective value of feasible solution}) - (\text{objective bound})|}{\max(\delta, |(\text{objective value of feasible solution})|)}.$$

where δ is given by the paramater MSK_DPAR_MIO_REL_GAP_CONST. Otherwise it has the value -1.0.

MSK_DINF_MIO_OPTIMIZER_TIME

Time spent in the optimizer while solving the relaxtions.

MSK_DINF_MIO_PROBING_TIME

Total time for probing.

MSK_DINF_MIO_ROOT_CUTGEN_TIME

Total time for cut generation.

MSK_DINF_MIO_ROOT_OPTIMIZER_TIME

Time spent in the optimizer while solving the root relaxation.

MSK_DINF_MIO_ROOT_PRESOLVE_TIME

Time spent in while presolveing the root relaxation.

MSK_DINF_MIO_TIME

Time spent in the mixed-integer optimizer.

MSK_DINF_MIO_USER_OBJ_CUT

If the objective cut is used, then this information item has the value of the cut.

MSK_DINF_OPTIMIZER_TIME

Total time spent in the optimizer since it was invoked.

MSK_DINF_PRESOLVE_ELI_TIME

Total time spent in the eliminator since the presolve was invoked.

MSK_DINF_PRESOLVE_LINDEP_TIME

Total time spent in the linear dependency checker since the presolve was invoked.

MSK_DINF_PRESOLVE_TIME

Total time (in seconds) spent in the presolve since it was invoked.

MSK_DINF_PRIMAL_REPAIR_PENALTY_OBJ

The optimal objective value of the penalty function.

MSK_DINF_QCQO_REFORMULATE_TIME

Time spent with conic quadratic reformulation.

MSK_DINF_RD_TIME

Time spent reading the data file.

MSK_DINF_SIM_DUAL_TIME

Time spent in the dual simplex optimizer since invoking it.

MSK_DINF_SIM_FEAS

Feasibility measure reported by the simplex optimizer.

MSK_DINF_SIM_NETWORK_DUAL_TIME

Time spent in the dual network simplex optimizer since invoking it.

MSK_DINF_SIM_NETWORK_PRIMAL_TIME

Time spent in the primal network simplex optimizer since invoking it.

MSK_DINF_SIM_NETWORK_TIME

Time spent in the network simplex optimizer since invoking it.

MSK_DINF_SIM_OBJ

Objective value reported by the simplex optimizer.

MSK_DINF_SIM_PRIMAL_DUAL_TIME

Time spent in the primal-dual simplex optimizer optimizer since invoking it.

MSK_DINF_SIM_PRIMAL_TIME

Time spent in the primal simplex optimizer since invoking it.

MSK_DINF_SIM_TIME

Time spent in the simplex optimizer since invoking it.

MSK_DINF_SOL_BAS_DUAL_OBJ

Dual objective value of the basic solution.

MSK_DINF_SOL_BAS_DVIOLCON

Maximal dual bound violation for x^c in the basic solution.

MSK_DINF_SOL_BAS_DVIOLVAR

Maximal dual bound violation for x^x in the basic solution.

MSK_DINF_SOL_BAS_PRIMAL_OBJ

Primal objective value of the basic solution.

MSK_DINF_SOL_BAS_PVIOLCON

Maximal primal bound violation for x^c in the basic solution.

MSK_DINF_SOL_BAS_PVIOLVAR

Maximal primal bound violation for x^x in the basic solution.

MSK_DINF_SOL_ITG_PRIMAL_OBJ

Primal objective value of the integer solution.

MSK_DINF_SOL_ITG_PVIOLBARVAR

Maximal primal bound violation for $\bar{X}$ in the integer solution.

MSK_DINF_SOL_ITG_PVIOLCON

Maximal primal bound violation for x^c in the integer solution.

MSK_DINF_SOL_ITG_PVIOLCONES

Maximal primal violation for primal conic constraints in the integer solution.

MSK_DINF_SOL_ITG_PVIOLITG

Maximal violation for the integer constraints in the integer solution.

MSK_DINF_SOL_ITG_PVIOLVAR

Maximal primal bound violation for x^x in the integer solution.

MSK_DINF_SOL_ITR_DUAL_OBJ

Dual objective value of the interior-point solution.

MSK_DINF_SOL_ITR_DVIOLBARVAR

Maximal dual bound violation for $\bar{X}$ in the interior-point solution.

MSK_DINF_SOL_ITR_DVIOLCON

Maximal dual bound violation for x^c in the interior-point solution.

MSK_DINF_SOL_ITR_DVIOLCONES

Maximal dual violation for dual conic constraints in the interior-point solution.

MSK_DINF_SOL_ITR_DVIOLVAR

Maximal dual bound violation for x^x in the interior-point solution.

MSK_DINF_SOL_ITR_PRIMAL_OBJ

Primal objective value of the interior-point solution.

MSK_DINF_SOL_ITR_PVIOLBARVAR

Maximal primal bound violation for $\bar{X}$ in the interior-point solution.

MSK_DINF_SOL_ITR_PVIOLCON

Maximal primal bound violation for x^c in the interior-point solution.

MSK_DINF_SOL_ITR_PVIOLCONES

Maximal primal violation for primal conic constraints in the interior-point solution.

MSK_DINF_SOL_ITR_PVIOLVAR

Maximal primal bound violation for x^x in the interior-point solution.

11.11 Feasibility repair types

MSK_FEASREPAIR_OPTIMIZE_NONE

Do not optimize the feasibility repair problem.

MSK_FEASREPAIR_OPTIMIZE_PENALTY

Minimize weighted sum of violations.

MSK_FEASREPAIR_OPTIMIZE_COMBINED

Minimize with original objective subject to minimal weighted violation of bounds.

11.12 License feature

MSK_FEATURE_PTS

Base system.

MSK_FEATURE_PTON

Nonlinear extension.

MSK_FEATURE_PTOM

Mixed-integer extension.

MSK_FEATURE_PTOX

Non-convex extension.

11.13 Integer information items.

MSK_IINF_ANA_PRO_NUM_CON

Number of constraints in the problem.

MSK_IINF_ANA_PRO_NUM_CON_EQ

Number of equality constraints.

MSK_IINF_ANA_PRO_NUM_CON_FR

Number of unbounded constraints.

MSK_IINF_ANA_PRO_NUM_CON_LO

Number of constraints with a lower bound and an infinite upper bound.

MSK_IINF_ANA_PRO_NUM_CON_RA

Number of constraints with finite lower and upper bounds.

MSK_IINF_ANA_PRO_NUM_CON_UP

Number of constraints with an upper bound and an infinite lower bound.

MSK_IINF_ANA_PRO_NUM_VAR

Number of variables in the problem.

MSK_IINF_ANA_PRO_NUM_VAR_BIN

Number of binary (0-1) variables.

MSK_IINF_ANA_PRO_NUM_VAR_CONT

Number of continuous variables.

MSK_IINF_ANA_PRO_NUM_VAR_EQ

Number of fixed variables.

MSK_IINF_ANA_PRO_NUM_VAR_FR

Number of free variables.

MSK_IINF_ANA_PRO_NUM_VAR_INT

Number of general integer variables.

MSK_IINF_ANA_PRO_NUM_VAR_LO

Number of variables with a lower bound and an infinite upper bound.

MSK_IINF_ANA_PRO_NUM_VAR_RA

Number of variables with finite lower and upper bounds.

MSK_IINF_ANA_PRO_NUM_VAR_UP

Number of variables with an upper bound and an infinite lower bound. This value is set by

`MSK_IINF_CONCURRENT_FASTEST_OPTIMIZER`

The type of the optimizer that finished first in a concurrent optimization.

`MSK_IINF_INTPNT_FACTOR_DIM_DENSE`

Dimension of the dense sub system in factorization.

`MSK_IINF_INTPNT_ITER`

Number of interior-point iterations since invoking the interior-point optimizer.

`MSK_IINF_INTPNT_NUM_THREADS`

Number of threads that the interior-point optimizer is using.

`MSK_IINF_INTPNT_SOLVE_DUAL`

Non-zero if the interior-point optimizer is solving the dual problem.

`MSK_IINF_MIO_CONSTRUCT_NUM_ROUNDINGS`

Number of values in the integer solution that is rounded to an integer value.

`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.

`MSK_IINF_MIO_INITIAL_SOLUTION`

Is non-zero if an initial integer solution is specified.

`MSK_IINF_MIO_NUM_ACTIVE_NODES`

Number of active brabch bound nodes.

`MSK_IINF_MIO_NUM_BASIS_CUTS`

Number of basis cuts.

`MSK_IINF_MIO_NUM_BRANCH`

Number of branches performed during the optimization.

`MSK_IINF_MIO_NUM_CARDGUB_CUTS`

Number of cardgub cuts.

`MSK_IINF_MIO_NUM_CLIQUE_CUTS`

Number of clique cuts.

`MSK_IINF_MIO_NUM_COEF_REDC_CUTS`

Number of coef. redc. cuts.

`MSK_IINF_MIO_NUM_CONTRA_CUTS`

Number of contra cuts.

MSK_IINF_MIO_NUM_DISAGG_CUTS

Number of diasagg cuts.

MSK_IINF_MIO_NUM_FLOW_COVER_CUTS

Number of flow cover cuts.

MSK_IINF_MIO_NUM_GCD_CUTS

Number of gcd cuts.

MSK_IINF_MIO_NUM_GOMORY_CUTS

Number of Gomory cuts.

MSK_IINF_MIO_NUM_GUB_COVER_CUTS

Number of GUB cover cuts.

MSK_IINF_MIO_NUM_INT_SOLUTIONS

Number of integer feasible solutions that has been found.

MSK_IINF_MIO_NUM_KNAPSUR_COVER_CUTS

Number of knapsack cover cuts.

MSK_IINF_MIO_NUM_LATTICE_CUTS

Number of lattice cuts.

MSK_IINF_MIO_NUM_LIFT_CUTS

Number of lift cuts.

MSK_IINF_MIO_NUM_OBJ_CUTS

Number of obj cuts.

MSK_IINF_MIO_NUM_PLAN_LOC_CUTS

Number of loc cuts.

MSK_IINF_MIO_NUM_RELAX

Number of relaxations solved during the optimization.

MSK_IINF_MIO_NUMCON

Number of constraints in the problem solved be the mixed-integer optimizer.

MSK_IINF_MIO_NUMINT

Number of integer variables in the problem solved be the mixed-integer optimizer.

MSK_IINF_MIO_NUMVAR

Number of variables in the problem solved be the mixed-integer optimizer.

MSK_IINF_MIO_OBJ_BOUND_DEFINED

Non-zero if a valid objective bound has been found, otherwise zero.

MSK_IINF_MIO_TOTAL_NUM_CUTS

Total number of cuts generated by the mixed-integer optimizer.

MSK_IINF_MIO_USER_OBJ_CUT

If it is non-zero, then the objective cut is used.

MSK_IINF_OPT_NUMCON

Number of constraints in the problem solved when the optimizer is called.

MSK_IINF_OPT_NUMVAR

Number of variables in the problem solved when the optimizer is called

MSK_IINF_OPTIMIZE_RESPONSE

The reponse code returned by optimize.

MSK_IINF_RD_NUMBARVAR

Number of variables read.

MSK_IINF_RD_NUMCON

Number of constraints read.

MSK_IINF_RD_NUMCONE

Number of conic constraints read.

MSK_IINF_RD_NUMINTVAR

Number of integer-constrained variables read.

MSK_IINF_RD_NUMQ

Number of nonempty Q matrixes read.

MSK_IINF_RD_NUMVAR

Number of variables read.

MSK_IINF_RD_PROTYPE

Problem type.

MSK_IINF_SIM_DUAL_DEG_ITER

The number of dual degenerate iterations.

MSK_IINF_SIM_DUAL_HOTSTART

If 1 then the dual simplex algorithm is solving from an advanced basis.

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.

MSK_IINF_SIM_DUAL_INF_ITER

The number of iterations taken with dual infeasibility.

MSK_IINF_SIM_DUAL_ITER

Number of dual simplex iterations during the last optimization.

MSK_IINF_SIM_NETWORK_DUAL_DEG_ITER

The number of dual network degenerate iterations.

MSK_IINF_SIM_NETWORK_DUAL_HOTSTART

If 1 then the dual network simplex algorithm is solving from an advanced basis.

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.

MSK_IINF_SIM_NETWORK_DUAL_INF_ITER

The number of iterations taken with dual infeasibility in the network optimizer.

MSK_IINF_SIM_NETWORK_DUAL_ITER

Number of dual network simplex iterations during the last optimization.

MSK_IINF_SIM_NETWORK_PRIMAL_DEG_ITER

The number of primal network degenerate iterations.

MSK_IINF_SIM_NETWORK_PRIMAL_HOTSTART

If 1 then the primal network simplex algorithm is solving from an advanced basis.

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.

MSK_IINF_SIM_NETWORK_PRIMAL_INF_ITER

The number of iterations taken with primal infeasibility in the network optimizer.

MSK_IINF_SIM_NETWORK_PRIMAL_ITER

Number of primal network simplex iterations during the last optimization.

MSK_IINF_SIM_NUMCON

Number of constraints in the problem solved by the simplex optimizer.

MSK_IINF_SIM_NUMVAR

Number of variables in the problem solved by the simplex optimizer.

MSK_IINF_SIM_PRIMAL_DEG_ITER

The number of primal degenerate iterations.

MSK_IINF_SIM_PRIMAL_DUAL_DEG_ITER

The number of degenerate major iterations taken by the primal dual simplex algorithm.

MSK_IINF_SIM_PRIMAL_DUAL_HOTSTART

If 1 then the primal dual simplex algorithm is solving from an advanced basis.

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.

MSK_IINF_SIM_PRIMAL_DUAL_INF_ITER

The number of master iterations with dual infeasibility taken by the primal dual simplex algorithm.

MSK_IINF_SIM_PRIMAL_DUAL_ITER

Number of primal dual simplex iterations during the last optimization.

MSK_IINF_SIM_PRIMAL_HOTSTART

If 1 then the primal simplex algorithm is solving from an advanced basis.

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.

MSK_IINF_SIM_PRIMAL_INF_ITER

The number of iterations taken with primal infeasibility.

MSK_IINF_SIM_PRIMAL_ITER

Number of primal simplex iterations during the last optimization.

MSK_IINF_SIM_SOLVE_DUAL

Is non-zero if dual problem is solved.

MSK_IINF_SOL_BAS_PROSTA

Problem status of the basic solution. Updated after each optimization.

MSK_IINF_SOL_BAS_SOLSTA

Solution status of the basic solution. Updated after each optimization.

MSK_IINF_SOL_INT_PROSTA

Deprecated.

MSK_IINF_SOL_INT_SOLSTA

Degrecated.

MSK_IINF_SOL_ITG_PROSTA

Problem status of the integer solution. Updated after each optimization.

MSK_IINF_SOL_ITG_SOLSTA

Solution status of the integer solution. Updated after each optimization.

MSK_IINF_SOL_ITR_PROSTA

Problem status of the interior-point solution. Updated after each optimization.

MSK_IINF_SOL_ITR_SOLSTA

Solution status of the interior-point solution. Updated after each optimization.

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.

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.

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.

11.14 Information item types

MSK_INF_DOU_TYPE

Is a double information type.

MSK_INF_INT_TYPE

Is an integer.

MSK_INF_LINT_TYPE

Is a long integer.

11.15 Hot-start type employed by the interior-point optimizers.

MSK_INTPNT_HOTSTART_NONE

The interior-point optimizer performs a coldstart.

MSK_INTPNT_HOTSTART_PRIMAL

The interior-point optimizer exploits the primal solution only.

MSK_INTPNT_HOTSTART_DUAL

The interior-point optimizer exploits the dual solution only.

MSK_INTPNT_HOTSTART_PRIMAL_DUAL

The interior-point optimizer exploits both the primal and dual solution.

11.16 Input/output modes

MSK_IOMODE_READ

The file is read-only.

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.

MSK_IOMODE_READWRITE

The file is to read and written.

11.17 Language selection constants

MSK_LANG_ENG

English language selection

MSK_LANG_DAN

Danish language selection

11.18 Long integer information items.

MSK_LIINF_BI_CLEAN_DUAL_DEG_ITER

Number of dual degenerate clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_DUAL_ITER

Number of dual clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_PRIMAL_DEG_ITER

Number of primal degenerate clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_PRIMAL_DUAL_DEG_ITER

Number of primal-dual degenerate clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_PRIMAL_DUAL_ITER

Number of primal-dual clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_PRIMAL_DUAL_SUB_ITER

Number of primal-dual subproblem clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_PRIMAL_ITER

Number of primal clean iterations performed in the basis identification.

MSK_LIINF_BI_DUAL_ITER

Number of dual pivots performed in the basis identification.

MSK_LIINF_BI_PRIMAL_ITER

Number of primal pivots performed in the basis identification.

MSK_LIINF_INTPNT_FACTOR_NUM_NZ

Number of non-zeros in factorization.

MSK_LIINF_MIO_INTPNT_ITER

Number of interior-point iterations performed by the mixed-integer optimizer.

MSK_LIINF_MIO_SIMPLEX_ITER

Number of simplex iterations performed by the mixed-integer optimizer.

MSK_LIINF_RD_NUMANZ

Number of non-zeros in A that is read.

MSK_LIINF_RD_NUMQNZ

Number of Q non-zeros.

11.19 Mark

MSK_MARK_LO

The lower bound is selected for sensitivity analysis.

MSK_MARK_UP

The upper bound is selected for sensitivity analysis.

11.20 Continuous mixed-integer solution type

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

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_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.

11.21 Integer restrictions

MSK_MIO_MODE_IGNORED

The integer constraints are ignored and the problem is solved as a continuous problem.

MSK_MIO_MODE_SATISFIED

Integer restrictions should be satisfied.

MSK_MIO_MODE_LAZY

Integer restrictions should be satisfied if an optimizer is available for the problem.

11.22 Mixed-integer node selection types

MSK_MIO_NODE_SELECTION_FREE

The optimizer decides the node selection strategy.

MSK_MIO_NODE_SELECTION_FIRST

The optimizer employs a depth first node selection strategy.

MSK_MIO_NODE_SELECTION_BEST

The optimizer employs a best bound node selection strategy.

MSK_MIO_NODE_SELECTION_WORST

The optimizer employs a worst bound node selection strategy.

MSK_MIO_NODE_SELECTION_HYBRID

The optimizer employs a hybrid strategy.

MSK_MIO_NODE_SELECTION_PSEUDO

The optimizer employs selects the node based on a pseudo cost estimate.

11.23 MPS file format type

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.

11.24 Message keys

MSK_MSG_READING_FILE

MSK_MSG_WRITING_FILE

MSK_MSG_MPS_SELECTED

11.25 Name types

MSK_NAME_TYPE_GEN

General names. However, no duplicate and blank names are allowed.

MSK_NAME_TYPE_MPS

MPS type names.

MSK_NAME_TYPE_LP

LP type names.

11.26 Objective sense types

MSK_OBJECTIVE_SENSE_MINIMIZE

The problem should be minimized.

MSK_OBJECTIVE_SENSE_MAXIMIZE

The problem should be maximized.

11.27 On/off

MSK_OFF

Switch the option off.

MSK_ON

Switch the option on.

11.28 Optimizer types

MSK_OPTIMIZER_FREE

The optimizer is chosen automatically.

MSK_OPTIMIZER_INTPNT

The interior-point optimizer is used.

MSK_OPTIMIZER_CONIC

The optimizer for problems having conic constraints.

MSK_OPTIMIZER_PRIMAL_SIMPLEX

The primal simplex optimizer is used.

MSK_OPTIMIZER_DUAL_SIMPLEX

The dual simplex optimizer is used.

MSK_OPTIMIZER_PRIMAL_DUAL_SIMPLEX

The primal dual simplex optimizer is used.

MSK_OPTIMIZER_FREE_SIMPLEX

One of the simplex optimizers is used.

MSK_OPTIMIZER_NETWORK_PRIMAL_SIMPLEX

The network primal simplex optimizer is used. It is only applicable to pute network problems.

MSK_OPTIMIZER_MIXED_INT_CONIC

The mixed-integer optimizer for conic and linear problems.

MSK_OPTIMIZER_MIXED_INT

The mixed-integer optimizer.

MSK_OPTIMIZER_CONCURRENT

The optimizer for nonconvex nonlinear problems.

MSK_OPTIMIZER_NONCONVEX

The optimizer for nonconvex nonlinear problems.

11.29 Ordering strategies

MSK_ORDER_METHOD_FREE

The ordering method is chosen automatically.

MSK_ORDER_METHOD_APPMINLOC

Approximate minimum local fill-in ordering is employed.

MSK_ORDER_METHOD_EXPERIMENTAL

This option should not be used.

MSK_ORDER_METHOD_TRY_GRAPHPAR

Always try the the graph partitioning based ordering.

MSK_ORDER_METHOD_FORCE_GRAPHPAR

Always use the graph partitioning based ordering even if it is worse that the approximate minimum local fill ordering.

MSK_ORDER_METHOD_NONE

No ordering is used.

11.30 Parameter type

MSK_PAR_INVALID_TYPE

Not a valid parameter.

MSK_PAR_DOU_TYPE

Is a double parameter.

MSK_PAR_INT_TYPE

Is an integer parameter.

MSK_PAR_STR_TYPE

Is a string parameter.

11.31 Presolve method.

MSK_PRESOLVE_MODE_OFF

The problem is not presolved before it is optimized.

MSK_PRESOLVE_MODE_ON

The problem is presolved before it is optimized.

MSK_PRESOLVE_MODE_FREE

It is decided automatically whether to presolve before the problem is optimized.

11.32 Problem data items

MSK_PI_VAR

Item is a variable.

MSK_PI_CON

Item is a constraint.

MSK_PI_CONE

Item is a cone.

11.33 Problem types

MSK_PROBTYPE_LO

The problem is a linear optimization problem.

MSK_PROBTYPE_QO

The problem is a quadratic optimization problem.

MSK_PROBTYPE_QCQO

The problem is a quadratically constrained optimization problem.

MSK_PROBTYPE_GECO

General convex optimization.

MSK_PROBTYPE_CONIC

A conic optimization.

MSK_PROBTYPE_MIXED

General nonlinear constraints and conic constraints. This combination can not be solved by MOSEK.

11.34 Problem status keys

MSK_PRO_STA_UNKNOWN

Unknown problem status.

MSK_PRO_STA_PRIM_AND_DUAL_FEAS

The problem is primal and dual feasible.

MSK_PRO_STA_PRIM_FEAS

The problem is primal feasible.

MSK_PRO_STA_DUAL_FEAS

The problem is dual feasible.

MSK_PRO_STA_PRIM_INFEAS

The problem is primal infeasible.

MSK_PRO_STA_DUAL_INFEAS

The problem is dual infeasible.

MSK_PRO_STA_PRIM_AND_DUAL_INFEAS

The problem is primal and dual infeasible.

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.

MSK_PRO_STA_NEAR_PRIM_AND_DUAL_FEAS

The problem is at least nearly primal and dual feasible.

MSK_PRO_STA_NEAR_PRIM_FEAS

The problem is at least nearly primal feasible.

MSK_PRO_STA_NEAR_DUAL_FEAS

The problem is at least nearly dual feasible.

MSK_PRO_STA_PRIM_INFEAS_OR_UNBOUNDED

The problem is either primal infeasible or unbounded. This may occur for mixed-integer problems.

11.35 Response code type

MSK_RESPONSE_OK

The response code is OK.

MSK_RESPONSE_WRN

The response code is a warning.

MSK_RESPONSE_TRM

The response code is an optimizer termination status.

MSK_RESPONSE_ERR

The response code is an error.

MSK_RESPONSE_UNK

The response code does not belong to any class.

11.36 Scaling type

MSK_SCALING_METHOD_POW2

Scales only with power of 2 leaving the mantissa untouched.

MSK_SCALING_METHOD_FREE

The optimizer chooses the scaling heuristic.

11.37 Scaling type

MSK_SCALING_FREE

The optimizer chooses the scaling heuristic.

MSK_SCALING_NONE

No scaling is performed.

MSK_SCALING_MODERATE

A conservative scaling is performed.

MSK_SCALING_AGGRESSIVE

A very aggressive scaling is performed.

11.38 Sensitivity types

MSK_SENSITIVITY_TYPE_BASIS

Basis sensitivity analysis is performed.

MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION

Optimal partition sensitivity analysis is performed.

11.39 Degeneracy strategies

MSK_SIM_DEGEN_NONE

The simplex optimizer should use no degeneration strategy.

MSK_SIM_DEGEN_FREE

The simplex optimizer chooses the degeneration strategy.

MSK_SIM_DEGEN_AGGRESSIVE

The simplex optimizer should use an aggressive 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.

11.40 Exploit duplicate columns.

MSK_SIM_EXPLOIT_DUPVEC_OFF

Disallow the simplex optimizer to exploit duplicated columns.

MSK_SIM_EXPLOIT_DUPVEC_ON

Allow the simplex optimizer to exploit duplicated columns.

MSK_SIM_EXPLOIT_DUPVEC_FREE

The simplex optimizer can choose freely.

11.41 Hot-start type employed by the simplex optimizer

MSK_SIM_HOTSTART_NONE

The simplex optimizer performs a coldstart.

MSK_SIM_HOTSTART_FREE

The simplex optimize chooses the hot-start type.

MSK_SIM_HOTSTART_STATUS_KEYS

Only the status keys of the constraints and variables are used to choose the type of hot-start.

11.42 Problem reformulation.

MSK_SIM_REFORMULATION_OFF

Disallow the simplex optimizer to reformulate the problem.

MSK_SIM_REFORMULATION_ON

Allow the simplex optimizer to reformulate the problem.

MSK_SIM_REFORMULATION_FREE

The simplex optimizer can choose freely.

MSK_SIM_REFORMULATION_AGGRESSIVE

The simplex optimizer should use an aggressive reformulation strategy.

11.43 Simplex selection strategy

MSK_SIM_SELECTION_FREE

The optimizer chooses the pricing strategy.

MSK_SIM_SELECTION_FULL

The optimizer uses full pricing.

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).

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.

11.44 Solution items

MSK_SOL_ITEM_XC

Solution for the constraints.

MSK_SOL_ITEM_XX

Variable solution.

MSK_SOL_ITEM_Y

Lagrange multipliers for equations.

MSK_SOL_ITEM_SLC

Lagrange multipliers for lower bounds on the constraints.

MSK_SOL_ITEM_SUC

Lagrange multipliers for upper bounds on the constraints.

MSK_SOL_ITEM_SLX

Lagrange multipliers for lower bounds on the variables.

MSK_SOL_ITEM_SUX

Lagrange multipliers for upper bounds on the variables.

MSK_SOL_ITEM_SNX

Lagrange multipliers corresponding to the conic constraints on the variables.

11.45 Solution status keys

MSK_SOL_STA_UNKNOWN

Status of the solution is unknown.

MSK_SOL_STA_OPTIMAL

The solution is optimal.

MSK_SOL_STA_PRIM_FEAS

The solution is primal feasible.

MSK_SOL_STA_DUAL_FEAS

The solution is dual feasible.

MSK_SOL_STA_PRIM_AND_DUAL_FEAS

The solution is both primal and dual feasible.

MSK_SOL_STA_PRIM_INFEAS_CER

The solution is a certificate of primal infeasibility.

MSK_SOL_STA_DUAL_INFEAS_CER

The solution is a certificate of dual infeasibility.

MSK_SOL_STA_NEAR_OPTIMAL

The solution is nearly optimal.

MSK_SOL_STA_NEAR_PRIM_FEAS

The solution is nearly primal feasible.

MSK_SOL_STA_NEAR_DUAL_FEAS

The solution is nearly dual feasible.

MSK_SOL_STA_NEAR_PRIM_AND_DUAL_FEAS

The solution is nearly both primal and dual feasible.

MSK_SOL_STA_NEAR_PRIM_INFEAS_CER

The solution is almost a certificate of primal infeasibility.

MSK_SOL_STA_NEAR_DUAL_INFEAS_CER

The solution is almost a certificate of dual infeasibility.

MSK_SOL_STA_INTEGER_OPTIMAL

The primal solution is integer optimal.

MSK_SOL_STA_NEAR_INTEGER_OPTIMAL

The primal solution is near integer optimal.

11.46 Solution types

MSK_SOL_ITR

The interior solution.

MSK_SOL_BAS

The basic solution.

MSK_SOL_ITG

The integer solution.

11.47 Solve primal or dual form

MSK_SOLVE_FREE

The optimizer is free to solve either the primal or the dual problem.

MSK_SOLVE_PRIMAL

The optimizer should solve the primal problem.

MSK_SOLVE_DUAL

The optimizer should solve the dual problem.

11.48 Status keys

MSK_SK_UNK

The status for the constraint or variable is unknown.

MSK_SK_BAS

The constraint or variable is in the basis.

MSK_SK_SUPBAS

The constraint or variable is super basic.

MSK_SK_LOW

The constraint or variable is at its lower bound.

MSK_SK_UPR

The constraint or variable is at its upper bound.

MSK_SK_FIX

The constraint or variable is fixed.

MSK_SK_INF

The constraint or variable is infeasible in the bounds.

11.49 Starting point types

MSK_STARTING_POINT_FREE

The starting point is chosen automatically.

MSK_STARTING_POINT_GUESS

The optimizer guesses a starting point.

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_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.

11.50 Stream types

MSK_STREAM_LOG

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.

MSK_STREAM_MSG

Message stream. Log information relating to performance and progress of the optimization is written to this stream.

MSK_STREAM_ERR

Error stream. Error messages are written to this stream.

MSK_STREAM_WRN

Warning stream. Warning messages are written to this stream.

11.51 Symmetric matrix types

MSK_SYMMAT_TYPE_SPARSE

Sparse symmetric matrix.

11.52 Transposed matrix.

MSK_TRANSPOSE_NO

No transpose is applied.

MSK_TRANSPOSE_YES

A transpose is applied.

11.53 Triangular part of a symmetric matrix.

MSK_UPLO_LO

Lower part.

MSK_UPLO_UP

Upper part

11.54 Integer values

MSK_LICENSE_BUFFER_LENGTH

The length of a license key buffer.

MSK_MAX_STR_LEN

Maximum string length allowed in MOSEK.

11.55 Variable types

MSK_VAR_TYPE_CONT

Is a continuous variable.

MSK_VAR_TYPE_INT

Is an integer variable.

11.56 XML writer output mode

MSK_WRITE_XML_MODE_ROW

Write in row order.

MSK_WRITE_XML_MODE_COL

Write in column order.

Chapter 12

MOSEK Command line tool

12.1 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.

12.2 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.

12.3 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 9.

12.3.1 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
```

Chapter 13

The MPS file format

MOSEK supports the standard MPS format with some extensions. For a detailed description of the MPS format see the book by Nazareth [15].

13.1 MPS file structure

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

$$
\begin{array}{ccccc}
l^c & \leq & Ax + q(x) & \leq & u^c, \\
l^x & \leq & x & \leq & u^x, \\
 & & x \in \mathcal{C}, & & \\
 & & x_{\mathcal{J}} \text{ integer}, & &
\end{array}
\tag{13.1}
$$

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,

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

  where it is assumed that

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

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 13.5 for details.

13.1.1 Linear example `lo1.mps`

A concrete example of a MPS file is presented below:

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

Subsequently each individual section in the MPS format is discussed.

13.1.2 `NAME`

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

13.1.3 `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.

13.1.4 `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.

13.1.5 `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` with the 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 .

13.1.6 `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	Re-quired	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.

13.1.7 RHS (optional)

A record in this section has the format

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

where the requirements for each field are as follows:

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

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

13.1.8 `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 and `[cname1]` is a valid constraint name. Assume that `[cname1]` is assigned to the i th constraint and let v_1 be the value specified by `[value1]`, then a record has the interpretation:

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

13.1.9 `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	Re-quired	Description
`[vname1]`	5	8	Yes	Variable name
`[vname2]`	15	8	Yes	Variable name
`[value1]`	25	12	Yes	Numerical value
`[vname3]`	40	8	No	Variable name
`[value2]`	50	12	No	Numerical value

A record specifies one or two elements in the lower triangular part of the Q^i matrix where `[cname1]` specifies the i . Hence, if the names `[vname1]` and `[vname2]` have been assigned to the k th and j th variable, then Q^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

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

Regarding the `QSECTION`s please note that:

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

13.1.10 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	Re-quired	Description
??	2	2	Yes	Bound key
[name]	5	8	Yes	Name of the BOUNDS vector
[vname1]	15	8	Yes	Variable name
[value1]	25	12	No	Numerical value

Hence, a record in the BOUNDS section has the following interpretation: [name] is the name of the bound vector and [vname1] is the name of the variable 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	$\lceil v_1 \rceil$	*unchanged*	Yes
UI	unchanged	$\lfloor v_1 \rfloor$	Yes

v_1 is the value specified by [value1].

13.1.11 CSECTION (optional)

The purpose of the CSECTION is to specify the constraint

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

in (13.1).

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:

$$\mathcal{C}_t = \left\{ x \in \mathbb{R}^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\}. \tag{13.2}$$

- Rotated quadratic cone:

$$\mathcal{C}_t = \left\{ x \in \mathbb{R}^{n^t} : 2x_1 x_2 \geq \sum_{j=3}^{n^t} x_j^2, \ x_1, x_2 \geq 0 \right\}. \tag{13.3}$$

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

$$x_4 \geq \sqrt{x_5^2 + x_8^2} \tag{13.4}$$

and the rotated quadratic cone

$$2x_3 x_7 \geq x_1^2 + x_0^2, \ x_3, x_7 \geq 0, \tag{13.5}$$

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 (13.4) 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 (13.5). 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. (13.2).
RQUAD	≥ 2	Rotated quadratic cone i.e. (13.3).

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	Required	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.

13.1.12 ENDATA

This keyword denotes the end of the MPS file.

13.2 Integer variables

Using special bound keys in the BOUNDS section it is possible to specify that some or all of the variables should be integer-constrained i.e. be members of $\mathcal{J}$. However, an alternative method is available.

This method is available only for backward compatibility and we recommend that it is not used. This method requires that markers are placed in the COLUMNS section as in the example:

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

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

- 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.

13.3 General limitations

- An MPS file should be an ASCII file.

13.4 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.

13.5 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.

Chapter 14

The LP file format

MOSEK supports the LP file format with some extensions i.e. MOSEK can read and write LP formatted files.

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

The LP file format can specify problems 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

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

- $A \in \mathbb{R}^{m \times n}$ 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

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

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

14.1 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.

14.1.1 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

```
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

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

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.

14.1.2 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.

$$-5 \leq x_1 + x_2 \leq 5 \tag{14.1}$$

may be written as

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

By default MOSEK writes ranged constraints this way.

If the files must adhere to the LP standard, ranged constraints must either be split into upper bounded and lower bounded constraints or be written as en equality with a slack variable. For example the expression (14.1) may be written as

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

14.1.3 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
```

14.1.4 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.

14.1.5 Terminating section

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

```
end
```

14.1.6 Linear example `lo1.lp`

A simple example of an LP file is:

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

14.1.7 Mixed integer example `milo1.lp`

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

14.2 LP format peculiarities

14.2.1 Comments

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

14.2.2 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.

MOSEK accepts any character as valid for names, except `'\0'`. When writing a name that is not allowed in LP files, it is changed and a warning is issued.

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

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

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

14.2.3 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.

14.2.4 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.

14.3 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 with the definitions of other vendors, use the parameter 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 parameter setting

MSK_IPAR_WRITE_GENERIC_NAMES = MSK_ON

which will cause all names to be renamed systematically in the output file.

14.4 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.

14.4.1 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

14.4.2 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).

Chapter 15

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.

15.1 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.

15.2 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]
```

15.2.1 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 specified which has the same name as a previously defined constraint, the new constraint replaces the existing one.

- `[bounds]` This does not directly contain any data, but may contain 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

    $$-10 \leq x, y \leq 10.$$

  - `[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 of linear non-zeros in constraints) and `numqnz` (number of quadratic non-zeros in constraints).

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

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

  Note that a `[solution]`-section must be always specified inside a `[solutions]`-section. The syntax of a `[solution]`-section is the following:

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

  where `SOLTYPE` is one of the strings

  - ‘`interior`’, a non-basic solution,
  - ‘`basic`’, a basic solution,
  - ‘`integer`’, an integer solution,

  and `STATUS` is one of the strings

  - ‘UNKNOWN’,
  - ‘OPTIMAL’,
  - ‘INTEGER_OPTIMAL’,
  - ‘PRIM_FEAS’,
  - ‘DUAL_FEAS’,
  - ‘PRIM_AND_DUAL_FEAS’,
  - ‘NEAR_OPTIMAL’,
  - ‘NEAR_PRIM_FEAS’,
  - ‘NEAR_DUAL_FEAS’,
  - ‘NEAR_PRIM_AND_DUAL_FEAS’,
  - ‘PRIM_INFEAS_CER’,
  - ‘DUAL_INFEAS_CER’,
  - ‘NEAR_PRIM_INFEAS_CER’,

  - 'NEAR_DUAL_INFEAS_CER',
  - 'NEAR_INTEGER_OPTIMAL'.

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

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

```
KEYWORD=value
```

  Allowed keywords are as follows:

  - sk. The status of the item, where the value is one of the following strings:
    - LOW, the item is on its lower bound.
    - UPR, the item is on its upper bound.
    - FIX, it is a fixed item.
    - BAS, the item is in the basis.
    - SUPBAS, the item is super basic.
    - UNK, the status is unknown.
    - INF, the item is outside its bounds (infeasible).
  - lvl Defines the level of the item.
  - sl Defines the level of the dual variable associated with its lower bound.
  - su Defines the level of the dual variable associated with its upper bound.
  - sn Defines the level of the variable associated with its cone.
  - y Defines the level of the corresponding dual variable (for constraints only).

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

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

  An example of a solution section

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

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

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

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

15.2.2 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]+)?
```

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

15.3 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]
```

15.4 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.

15.5 Examples

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

15.5.1 Linear example lo1.opf

Consider the example:

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

having the bounds

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

In the `OPF` format the example is displayed as shown below:

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

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

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

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

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

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

15.5.2 Quadratic example `qo1.opf`

An example of a quadratic optimization problem is

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

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

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

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

```
[/hints]

[variables disallow_new_variables]
  x1 x2 x3
[/variables]

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

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

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

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

15.5.3 Conic quadratic example `cqo1.opf`

Consider the example:

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

Please note that the type of the cones is defined by the parameter to `[cone ...]`; the content of the `cone`-section is the names of variables that belong to the cone.

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

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

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

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

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

```
[/constraints]

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

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

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

  # Define rotated quadratic cone: 2 x5 x6 >= x3^2
  [cone rquad 'k2'] x5, x6, x3 [/cone]
[/bounds]
```

15.5.4 Mixed integer example `milo1.opf`

Consider the mixed integer problem:

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

This can be implemented in `OPF` with:

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

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

[variables disallow_new_variables]
  x1 x2
[/variables]

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

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

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

[integer]
```

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

Chapter 16

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.

Chapter 17

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.

17.1 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

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 17.1: Status keys.

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

`ACTIVITY`

Given the i th constraint on the form

$$l_i^c \leq \sum_{j=1}^{n} a_{ij} x_j \leq u_i^c, \tag{17.1}$$

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 (17.1)).

`UPPER LIMIT`

Is the quantity u_i^c (see (17.1)).

`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.

17.2 The integer solution file

The integer solution is equivalent to the basic and interior solution files except that no dual information is included.

Chapter 18

Problem analyzer examples

This appendix presents a few examples of the output produced by the problem analyzer described in Section 7.1. The first two problems are taken from the MIPLIB 2003 collection, http://miplib.zib.de/.

18.1 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

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

18.2 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

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

18.3 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

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

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

[2] MOSEK ApS. MOSEK Modeling manual, 2012. Last revised January 31 2013. http://docs.mosek.com/generic/modeling-a4.pdf

[3] Andersen, E. D. and Andersen, K. D.. Presolving in linear programming. Math. Programming 2:221-245

[4] Andersen, E. D., Gondzio, J., Mészáros, Cs. and Xu, X.. Implementation of interior point methods for large scale linear programming, Interior-point methods of mathematical programming p. 189-252, 1996. Kluwer Academic Publishers

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

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

[7] Ahuja, R. K., Magnanti, T. L. and Orlin, J. B.. Network flows, Optimization, vol. 1 p. 211-369, 1989. North Holland, Amsterdam

[8] Andersen, E. D., Roos, C. and Terlaky, T.. On implementing a primal-dual interior-point method for conic quadratic optimization. Math. Programming February 2

[9] Andersen, E. D. and Ye, Y.. A computational study of the homogeneous algorithm for large-scale convex optimization. Computational Optimization and Applications 10:243-269

[10] Andersen, E. D. and Ye, Y.. On a homogeneous algorithm for the monotone complementarity problem. Math. Programming February 2:375-399

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

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

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

[14] Wallace, S. W.. Decision making under uncertainty: Is sensitivity of any use. Oper. Res. January 1:20-25

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

Index