

**The MOSEK optimization toolbox for
MATLAB manual.
Version 6.0 (Revision 148).**



www.mosek.com

Published by MOSEK ApS, Denmark.

Copyright (c) 1998-2012 MOSEK ApS, Denmark. All rights reserved..

Disclaimer: MOSEK ApS (the author of MOSEK) accepts no responsibility for damages resulting from the use of the MOSEK software and makes no warranty, neither expressed nor implied, including, but not limited to, any implied warranty of fitness for a particular purpose. The software is provided as it is, and you, its user, assume all risks when using it.

Contact information

Phone +45 3917 9907

Fax +45 3917 9823

WEB <http://www.mosek.com>

Email sales@mosek.com
support@mosek.com
info@mosek.com

Sales, pricing, and licensing.

Technical support, questions and bug reports.

Everything else.

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

Contents

1	Changes and new features in MOSEK	3
1.1	Compilers used to build MOSEK	3
1.2	General changes	3
1.3	Optimizers	4
1.3.1	Interior point optimizer	4
1.3.2	The simplex optimizers	4
1.3.3	Mixed-integer optimizer	4
1.4	API changes	4
1.5	License system	5
1.6	Other changes	5
1.7	Interfaces	5
1.8	Platform changes	5
2	Introduction	7
2.1	What is optimization?	7
2.2	Why you need the MOSEK optimization toolbox	7
2.2.1	Features of the MOSEK optimization toolbox	7
2.3	Comparison with the MATLAB optimization toolbox	8
3	Supported MATLAB versions	11
4	Installation	13
4.1	Locating the toolbox functions	13
4.1.1	On Windows	13
4.1.2	On Linux/UNIX/MAC OS X	14
4.1.3	Permanently changing <code>matlabpath</code>	14
4.2	Verifying that MOSEK works	14
4.3	Troubleshooting	15
4.3.1	??? Undefined function or variable 'mosekopt'	15
4.3.2	"libgcc_s.so.1 must be installed for pthread_cancel to work"	16
4.3.3	Using the MATLAB compiler	16

4.3.4	Shadowing the m-file	17
4.3.5	“Cannot find authentication file”	17
5	Getting support and help	19
5.1	MOSEK documentation	19
5.2	Additional reading	19
6	MOSEK / MATLAB integration	21
6.1	MOSEK replacements for MATLAB functions	21
6.2	The license system	21
6.2.1	Waiting for a free license	22
7	A guided tour	23
7.1	Introduction	23
7.2	The tour starts	23
7.3	The MOSEK terminology	24
7.4	Linear optimization	24
7.4.1	Using <code>msklpopt</code>	25
7.4.2	Using <code>mosekopt</code>	26
7.5	Convex quadratic optimization	27
7.5.1	Two important assumptions	28
7.5.2	Using <code>mskqpopt</code>	28
7.5.3	Using <code>mosekopt</code>	29
7.6	Conic optimization	30
7.6.1	The conic optimization problem	30
7.6.2	Solving an example	31
7.6.3	Quadratic and conic optimization	33
7.6.4	Conic duality and the dual solution	34
7.6.5	Setting accuracy parameters for the conic optimizer	36
7.7	Quadratically constrained optimization	36
7.8	Linear least squares and related norm minimization problems	37
7.8.1	The case of the 2 norm	37
7.8.2	The case of the infinity norm	39
7.8.3	The case of the 1-norm	40
7.9	More about solving linear least squares problems	41
7.9.1	Using conic optimization on linear least squares problems	45
7.10	Entropy optimization	46
7.10.1	Using <code>mskenopt</code>	46
7.11	Geometric optimization	47
7.11.1	Using <code>mskgpopt</code>	48
7.11.2	Comments	49

7.12	Separable convex optimization	50
7.12.1	Using <code>mskscopt</code>	51
7.13	Mixed-integer optimization	53
7.13.1	Solving an example	54
7.13.2	Speeding up the solution of a mixed-integer problem	55
7.14	Sensitivity analysis	57
7.15	Inspecting a problem	59
7.16	The solutions	61
7.16.1	The constraint and variable status keys	62
7.17	Viewing the task information	62
7.18	Inspecting and setting parameters	63
7.19	Advanced start (hot-start)	64
7.19.1	Some examples using hot-start	64
7.19.2	Adding a new variable	65
7.19.3	Fixing a variable	66
7.19.4	Adding a new constraint	66
7.19.5	Using numeric values to represent status key codes	67
7.20	Using names	69
7.20.1	Blanks in names	69
7.21	MPS files	69
7.21.1	Reading an MPS file	69
7.21.2	Writing a MPS files	70
7.22	User call-back functions	71
7.22.1	Log printing via call-back function	71
7.22.2	The iteration call-back function	72
7.23	The license system	73
8	Command reference	75
8.1	Data structures	75
8.1.1	<code>prob</code>	75
8.1.2	<code>names</code>	80
8.1.3	<code>cones</code>	80
8.1.4	<code>sol</code>	81
8.1.5	<code>prisen</code>	81
8.1.6	<code>duasen</code>	82
8.1.7	<code>info</code>	83
8.1.8	<code>symbcon</code>	83
8.1.9	<code>callback</code>	83
8.2	An example of a command reference	83
8.3	Functions provided by the MOSEK optimization toolbox	84
8.4	MATLAB optimization toolbox compatible functions	88

8.4.1	Linear and quadratic optimization	88
8.4.2	For linear least squares problems	91
8.4.3	The optimization options	94
9	Case studies	97
9.1	Robust linear optimization	97
9.1.1	Introductory example	97
9.1.2	Data uncertainty and its consequences.	99
9.1.3	Robust linear optimization methodology	101
9.1.4	Random uncertainty and ellipsoidal robust counterpart	105
9.1.5	Further references	113
9.2	Geometric (posynomial) optimization	114
9.2.1	The problem	114
9.2.2	Applications	115
9.2.3	Modeling tricks	115
9.2.4	Problematic formulations	116
9.2.5	An example	117
9.2.6	Solving the example	117
9.2.7	Exporting to a file	118
9.2.8	Further information	119
10	Modelling	121
10.1	Linear optimization	121
10.1.1	Duality for linear optimization	122
10.1.2	Primal and dual infeasible case	125
10.2	Quadratic and quadratically constrained optimization	125
10.2.1	A general recommendation	125
10.2.2	Reformulating as a separable quadratic problem	126
10.3	Conic optimization	127
10.3.1	Duality for conic optimization	128
10.3.2	Infeasibility	129
10.3.3	Examples	129
10.3.4	Potential pitfalls in conic optimization	136
10.4	Nonlinear convex optimization	138
10.4.1	Duality	139
10.5	Recommendations	140
10.5.1	Avoid near infeasible models	141
10.6	Examples continued	141
10.6.1	The absolute value	141
10.6.2	The Markowitz portfolio model	142

11 The optimizers for continuous problems	147
11.1 How an optimizer works	147
11.1.1 Presolve	147
11.1.2 Dualizer	149
11.1.3 Scaling	149
11.1.4 Using multiple CPU's	150
11.2 Linear optimization	150
11.2.1 Optimizer selection	150
11.2.2 The interior-point optimizer	150
11.2.3 The simplex based optimizer	155
11.2.4 The interior-point or the simplex optimizer?	157
11.2.5 The primal or the dual simplex variant?	157
11.3 Linear network optimization	157
11.3.1 Network flow problems	157
11.3.2 Embedded network problems	158
11.4 Conic optimization	158
11.4.1 The interior-point optimizer	158
11.5 Nonlinear convex optimization	159
11.5.1 The interior-point optimizer	159
11.6 Solving problems in parallel	160
11.6.1 Thread safety	160
11.6.2 The parallelized interior-point optimizer	161
11.6.3 The concurrent optimizer	161
11.7 Understanding solution quality	162
11.7.1 The solution summary	162
12 The optimizer for mixed integer problems	165
12.1 Some notation	165
12.2 An important fact about integer optimization problems	166
12.3 How the integer optimizer works	166
12.3.1 Presolve	167
12.3.2 Heuristic	167
12.3.3 The optimization phase	167
12.4 Termination criterion	167
12.5 How to speed up the solution process	169
12.6 Understanding solution quality	169
12.6.1 Solutionsummary	169

13 The analyzers	171
13.1 The problem analyzer	171
13.1.1 General characteristics	173
13.1.2 Objective	174
13.1.3 Linear constraints	174
13.1.4 Constraint and variable bounds	175
13.1.5 Quadratic constraints	175
13.1.6 Conic constraints	175
13.2 Analyzing infeasible problems	175
13.2.1 Example: Primal infeasibility	176
13.2.2 Locating the cause of primal infeasibility	177
13.2.3 Locating the cause of dual infeasibility	178
13.2.4 The infeasibility report	178
13.2.5 Theory concerning infeasible problems	182
13.2.6 The certificate of primal infeasibility	183
13.2.7 The certificate of dual infeasibility	183
14 Sensitivity analysis	185
14.1 Introduction	185
14.2 Restrictions	185
14.3 References	185
14.4 Sensitivity analysis for linear problems	186
14.4.1 The optimal objective value function	186
14.4.2 The basis type sensitivity analysis	187
14.4.3 The optimal partition type sensitivity analysis	188
14.4.4 An example	190
14.5 Sensitivity analysis in the MATLAB toolbox	192
14.5.1 On bounds	192
14.5.2 Selecting analysis type	194
14.5.3 An example	194
A The MPS file format	199
A.1 The MPS file format	199
A.1.1 An example	201
A.1.2 NAME	201
A.1.3 OBJSENSE (optional)	202
A.1.4 OBJNAME (optional)	202
A.1.5 ROWS	202
A.1.6 COLUMNS	203
A.1.7 RHS (optional)	203
A.1.8 RANGES (optional)	204

A.1.9	QSECTION (optional)	205
A.1.10	BOUNDS (optional)	206
A.1.11	CSECTION (optional)	207
A.1.12	ENDATA	209
A.2	Integer variables	210
A.3	General limitations	210
A.4	Interpretation of the MPS format	211
A.5	The free MPS format	211
B	The LP file format	213
B.1	A warning	213
B.2	The LP file format	213
B.2.1	The sections	214
B.2.2	LP format peculiarities	218
B.2.3	The strict LP format	219
B.2.4	Formatting of an LP file	219
C	The OPF format	221
C.1	Intended use	221
C.2	The file format	221
C.2.1	Sections	222
C.2.2	Numbers	227
C.2.3	Names	227
C.3	Parameters section	228
C.4	Writing OPF files from MOSEK	228
C.5	Examples	229
C.5.1	Linear example <code>lo1.opf</code>	229
C.5.2	Quadratic example <code>qo1.opf</code>	230
C.5.3	Conic quadratic example <code>cqo1.opf</code>	231
C.5.4	Mixed integer example <code>milo1.opf</code>	232
D	The XML (OSiL) format	233
E	Parameters	235
E.1	Parameter groups	235
E.1.1	Logging parameters.	235
E.1.2	Basis identification parameters.	237
E.1.3	The Interior-point method parameters.	237
E.1.4	Simplex optimizer parameters.	240
E.1.5	Primal simplex optimizer parameters.	242
E.1.6	Dual simplex optimizer parameters.	243

E.1.7	Network simplex optimizer parameters.	243
E.1.8	Nonlinear convex method parameters.	243
E.1.9	The conic interior-point method parameters.	244
E.1.10	The mixed-integer optimization parameters.	245
E.1.11	Presolve parameters.	247
E.1.12	Termination criterion parameters.	248
E.1.13	Progress call-back parameters.	250
E.1.14	Non-convex solver parameters.	251
E.1.15	Feasibility repair parameters.	251
E.1.16	Optimization system parameters.	251
E.1.17	Output information parameters.	252
E.1.18	Extra information about the optimization problem.	254
E.1.19	Overall solver parameters.	254
E.1.20	Behavior of the optimization task.	255
E.1.21	Data input/output parameters.	256
E.1.22	Analysis parameters.	262
E.1.23	Solution input/output parameters.	262
E.1.24	Infeasibility report parameters.	264
E.1.25	License manager parameters.	264
E.1.26	Data check parameters.	265
E.1.27	Debugging parameters.	266
E.2	Double parameters	266
E.3	Integer parameters	291
E.4	String parameter types	376
F	Symbolic constants	385
F.1	Constraint or variable access modes	385
F.2	Function opcode	385
F.3	Function operand type	386
F.4	Basis identification	386
F.5	Bound keys	387
F.6	Specifies the branching direction.	387
F.7	Progress call-back codes	387
F.8	Types of convexity checks.	396
F.9	Compression types	396
F.10	Cone types	396
F.11	CPU type	397
F.12	Data format types	397
F.13	Double information items	398
F.14	Double parameters	403
F.15	Feasibility repair types	409

F.16 License feature	410
F.17 Integer information items.	410
F.18 Information item types	416
F.19 Input/output modes	416
F.20 Integer parameters	417
F.21 Language selection constants	435
F.22 Long integer information items.	435
F.23 Mark	436
F.24 Continuous mixed-integer solution type	437
F.25 Integer restrictions	437
F.26 Mixed-integer node selection types	438
F.27 MPS file format type	438
F.28 Message keys	439
F.29 Network detection method	439
F.30 Objective sense types	439
F.31 On/off	439
F.32 Optimizer types	440
F.33 Ordering strategies	440
F.34 Parameter type	441
F.35 Presolve method.	441
F.36 Problem data items	442
F.37 Problem types	442
F.38 Problem status keys	442
F.39 Interpretation of quadratic terms in MPS files	443
F.40 Response codes	444
F.41 Response code type	465
F.42 Scaling type	466
F.43 Scaling type	466
F.44 Sensitivity types	466
F.45 Degeneracy strategies	466
F.46 Exploit duplicate columns.	467
F.47 Hot-start type employed by the simplex optimizer	467
F.48 Problem reformulation.	467
F.49 Simplex selection strategy	468
F.50 Solution items	468
F.51 Solution status keys	469
F.52 Solution types	470
F.53 Solve primal or dual form	470
F.54 String parameter types	471
F.55 Status keys	473
F.56 Starting point types	473

F.57 Stream types	474
F.58 Integer values	474
F.59 Variable types	474
F.60 XML writer output mode	475
G Problem analyzer examples	477
G.1 air04	477
G.2 arki001	478
G.3 Problem with both linear and quadratic constraints	480
G.4 Problem with both linear and conic constraints	482

License agreement

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

`mosek\6\license.pdf`

Chapter 1

Changes and new features in MOSEK

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

1.1 Compilers used to build MOSEK

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

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

Table 1.1: Compiler version used to build MOSEK

1.2 General changes

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

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

1.3 Optimizers

1.3.1 Interior point optimizer

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

1.3.2 The simplex optimizers

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

1.3.3 Mixed-integer optimizer

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

1.4 API changes

- The Mosek/Java API is now built for SUN Java 1.5 and later.
- The Mosek/.NET API is now built for MS .NET 2.0 and later.
- The Mosek/Python API is now based on Python CTypes and uses NumPy instead of Numeric. Python 2.5 and later is supported on all platforms where the `ctypes` module is available.

1.5 License system

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

1.6 Other changes

- The documentation has been improved.

1.7 Interfaces

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

1.8 Platform changes

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

Chapter 2

Introduction

This manual describes the features of the MOSEK optimization toolbox for MATLAB. The toolbox makes it possible to call the highly efficient MOSEK optimization engine from the MATLAB environment.

2.1 What is optimization?

Many decision problems facing individuals and companies can be cast as an optimization problem i.e. making an optimal decision given some constraints specifying the possible decisions. As an example consider the problem of determining an optimal production plan. This can be formulated as maximizing a profit function given a set of constraints specifying the possible production plans.

2.2 Why you need the MOSEK optimization toolbox

Before solving an optimization problem data is gathered and prepared. Subsequently an optimization problem is formulated based on this data and the problem is communicated to the optimization software. Finally, when the results have been obtained, they are analyzed and interpreted. A popular software tool for these tasks is MATLAB¹. The MOSEK optimization toolbox provides an industrial strength solver capable of solving huge problems that other less specialized MATLAB packages can't solve.

2.2.1 Features of the MOSEK optimization toolbox

Below is a partial list of features in the MOSEK optimization toolbox.

- Solve linear optimization problems using either an interior-point or a simplex optimizer.

¹MATLAB is made by MathWorks, see <http://www.mathworks.com>.

- Solve convex quadratic optimization problems.
- Handle convex quadratic constraints.
- Solve conic quadratic optimization problems.
- Solve mixed-integer linear optimization problems.
- Solve linear least squares problems. The problem can have arbitrary linear side constraints.
- Solve linear ℓ_1 and ℓ_∞ norm minimization problems.
- Solve linearly constrained entropy optimization problems.
- Solve geometric programming problems (posynomial programming).
- Solve separable convex optimization problems.
- Read and write industry standard MPS files.

2.3 Comparison with the MATLAB optimization toolbox

MathWorks, the maker of MATLAB, also sells an optimization toolbox so an obvious question is how these two products compares on the following issues²:

Problem types: The MOSEK optimization toolbox can solve only **convex** optimization problems whereas the MATLAB toolbox handles nonconvex problems too.

On the other hand the MOSEK optimization toolbox can solve linear, quadratic and conic mixed-integer optimization problems which is not possible using the MATLAB optimization toolbox.

Algorithms: The emphasize of the MOSEK optimization toolbox is on **large-scale and sparse** problems. MOSEK offers only large-scale algorithms, but these algorithms perform very well for small and medium-sized problems too.

The main computational engine within the MOSEK optimization toolbox is a primal-dual type interior-point algorithm which has been demonstrated to be very well-suited for solving large-scale problems. Particularly when the algorithm is implemented using state-of-the-art (sparse) linear algebra as is the case for the MOSEK optimization toolbox. Readers interested in further details are referred to [4].

Furthermore, a primal and a dual simplex optimizer is available for linear problems.

²This is based on version 2 of the MATLAB optimization toolbox.

Compatibility: The MOSEK optimization toolbox for MATLAB includes the following functions

- `linprog`
- `lsqlin`
- `lsqnonneg`
- `optimget`
- `optimset`
- `quadprog`

which are also available in the MATLAB optimization toolbox. Moreover, these functions are compatible with the MATLAB functions of the same name in the sense that they accept the same arguments and return the same information.

The only differences between the functionality of the MOSEK and the MATLAB version of these functions are that the MOSEK version does not use all the MATLAB options, does not use an optional starting point³, and the MOSEK version of `quadprog` is intended for convex problems only. On the other hand the large-scale version of the MATLAB optimization toolbox does not accept arbitrary bounds and linear side constraints for quadratic problems whereas MOSEK does.

In general, for problems that both the MATLAB and the MOSEK optimization toolboxes handles, MOSEK delivers better reliability and performance.

³The large-scale linear programming optimizer in MATLAB does not use an optional starting point either.

Chapter 3

Supported MATLAB versions

Table 3.1 shows on which platforms and for which MATLAB versions the MOSEK optimization toolbox is available.

Platform type	MATLAB version	
	R2006B-R2009A	R2009b
linux32x86	Yes	Yes
linux64x86	Yes	Yes
osx32x86	Yes	Yes
osx64x86		Yes
solaris32x86		
solaris64x86		
win32x86	Yes	Yes
win64x86	Yes	Yes

Table 3.1: Supported MATLAB versions.

Chapter 4

Installation

In order to use the MOSEK optimization toolbox for MATLAB, you must install the MOSEK optimization tools. Please see Chapter 2 in the MOSEK installation manual for details on how to install MOSEK. An online version is available at

<http://www.mosek.com/documentation/>

4.1 Locating the toolbox functions

By default MATLAB cannot locate the MOSEK optimization toolbox functions. Therefore you must execute the `addpath` command within MATLAB to change the so-called `matlabpath` appropriately. Indeed `matlabpath` should include a path to the MOSEK optimization toolbox functions. The next subsections show how to use `addpath`.

4.1.1 On Windows

If you are using Windows you should do

```
% For MATLAB 7.9 (R2009b) or any later version do
addpath 'c:\Program Files\mosek\6\toolbox\r2009b'
```

```
% For MATLAB 7.4 (R2007a) to MATLAB 7.8 (R2009a)
addpath 'c:\Program Files\mosek\6\toolbox\r2007a'
```

```
% For MATLAB 7.3 (R2006b) do
addpath 'c:\Program Files\mosek\6\toolbox\r2006b'
```

This assumes that you installed MOSEK at

```
c:\Program Files\
```

If this is not the case, you will have to change the path given to `addpath`.

4.1.2 On Linux/UNIX/MAC OS X

If you are using UNIX or a UNIX-like operating system you should do

```
% For MATLAB 7.9 (R2009b) or any later version do
addpath '/home/user/mosek/6/toolbox/r2009b'
```

```
% For MATLAB 7.4 (R2007a) to MATLAB 7.8 (R2009a)
addpath '/home/user/mosek/6/toolbox/r2007a'
```

```
% For MATLAB 7.3 (R2006b)
addpath '/home/user/mosek/6/toolbox/r2006b'
```

This assumes that MOSEK is installed at
/home/user

If this is not the case, you will have to change the path given to addpath.

4.1.3 Permanently changing matlabpath

Normally, you will have to enter the `addpath` command every time MATLAB is started. This can be avoided if the `addpath` command is added to

```
<matlab>toolbox\local\startup.m
```

where `<matlab>` is the MATLAB root directory. Alternatively the permanent modification of the MATLAB path can be performed using the

```
\File\Set Path
```

menu item.

4.2 Verifying that MOSEK works

You can verify that MOSEK works by executing

```
mosekopt
```

in MATLAB. You should get a message similar to this:

```
MOSEK Version 3.1.1.62 (Build date: Dec 16 2004 11:49:51)
Copyright (c) 1998-2004 MOSEK ApS, Denmark. WWW: http://www.mosek.com
```

MOSEK command summary.

```
[r,res]=mosekopt(cmd,prob,param,log)
```

If you do not get this message, please read Section 4.3.

4.3 Troubleshooting

4.3.1 ??? Undefined function or variable 'mosekopt'

If you get the MATLAB error message

```
??? Undefined function or variable 'mosekopt'
```

you have not set up the `matlabpath` correctly as described in Section 4.1.

4.3.1.1 Unable to load MEX-file

One reason can be that you are not adding the correct path to the `matlabpath`, e.g. you may be trying to use the MOSEK optimization toolbox build for MATLAB 7 in MATLAB 6.

The other possible reasons are discussed below.

- Windows: MATLAB prints an error message similar to this:

```
DLL load failed for mex file
c:\mosek\3\tools\toolbox\14sp3\mosekopt.dll The
specified procedure could not be found. ??? Invalid MEX-file
```

Most likely this problem is caused by the fact that MOSEK cannot load the MOSEK DLL which in turn is caused by the environment variable

`PATH`

not being appropriately set up.

Please consult the *MOSEK installation manual* to learn how to install MOSEK under Windows and how to set up the operating system variable `PATH`.

- MAC OS X: The `DYLD_LIBRARY_PATH` environment variable is not appropriately set up. Setting this variable can be tricky, in particular if you are invoking MATLAB by clicking on the MATLAB icon. In this case a file named

```
$HOME/.MacOSX/environment.plist
```

with a proper content should exist on your computer. Further details about `environment.plist` and how to install MOSEK under MAC OS X are available in the *MOSEK installation manual*.

- UNIX: MATLAB prints an error message similar to this:

```
Unable to load mex file:
/usr/local/mosek/4/toolbox/14sp3/mosekopt.mexglx.
libmosek.so.2.5: cannot open shared object file: No such file or
directory ??? Invalid MEX-file
```

The cause of the problem is that the shared library

```
libmosek.so.2.5
```

cannot be loaded. Normally this problem is caused by the fact that the OS environment variable

```
LD_LIBRARY_PATH
```

is not appropriately set up. Please note that LD_LIBRARY_PATH may have another name on some UNIX systems. Please consult the *MOSEK installation manual* to learn how to install MOSEK under UNIX.

4.3.2 “libgcc_s.so.1 must be installed for pthread_cancel to work”

This error is caused by the fact that an old version of the

```
libgcc_s.so.1
```

library is included in the MATLAB distribution. One method of solving this is to execute

```
export LD_PRELOAD=/usr/lib/libgcc_s.so
```

before running MATLAB.

Another workaround is to remove libgcc_s.so.1 from the MATLAB distribution. We suggest you rename

```
<matlab>sys/os/glnx86/libgcc_s.so.1
```

to

```
<matlab>sys/os/glnx86/BACKUP_libgcc_s.so.1.bak
```

which should solve the problem.

4.3.3 Using the MATLAB compiler

MATLAB scripts using MOSEK can be compiled with the MATLAB compiler. Below is a description of some possible errors and their solution.

4.3.4 Shadowing the m-file

If you encounter the error

The file

```
'/tools/mosek/4/toolbox/r14sp3/mosekopt.mexglx'  
appears to be a MEX-file. It shadows the M-file  
'/tools/mosek/4/toolbox/r14sp3/mosekopt.m'  
but will not execute properly at runtime, as it does not export a function  
named 'mexFunction.'  
??? Error executing mcc, return status = 1.
```

when compiling a MATLAB script using MOSEK, you must delete

```
c:\mosek\6\toolbox\<MATLABVERSION>\mosekopt.m
```

This should fix the compile error.

4.3.5 “Cannot find authentication file”

If you encounter the error

```
Cannot find authentication file  
'C:\mosek\4\toolbox\r2006b\mosekopt_mexw32.auth'.
```

```
??? Invalid MEX-file 'C:\mosek\4\toolbox\r2006b\mosekopt_mexw32': .
```

Try to remove any `addpath` commands from your code when compiling. Instead, specify the location of the MOSEK files with

```
-I c:\mosek\4\toolbox\r2006b
```

in the compile command.

Chapter 5

Getting support and help

5.1 MOSEK documentation

For an overview of the available MOSEK documentation please see

`mosek\6\help\index.html`

in the distribution.

5.2 Additional reading

In this manual it is assumed that the reader is familiar with mathematics and in particular mathematical optimization. Some introduction to linear programming is found in books such as “Linear programming” by Chvátal [16] or “Computer Solution of Linear Programs” by Nazareth [21]. For more theoretical aspects see e.g. “Nonlinear programming: Theory and algorithms” by Bazaraa, Shetty, and Sherali [11]. Finally, the book “Model building in mathematical programming” by Williams [25] provides an excellent introduction to modeling issues in optimization.

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

Chapter 6

MOSEK / MATLAB integration

In this chapter we provide some details concerning the integration of MOSEK in MATLAB. The information in this chapter is not strictly necessary for basic use of the MOSEK optimization toolbox for MATLAB. The novice user can safely skip to the next chapter.

6.1 MOSEK replacements for MATLAB functions

MOSEK provides replacements for the MATLAB functions:

- `linprog`
- `quadprog`
- `optimget`
- `optimset`
- `lsqlin`
- `lsqnonneg`

The corresponding MATLAB file for each function is located in the `toolbox/solvers` directory of the MOSEK distribution. To use the MATLAB version of these functions instead of the MOSEK version, delete the MATLAB files provided by MOSEK.

6.2 The license system

By default a license token remains checked out for the duration of the MATLAB session. This can be changed such that the license is returned after each call to MOSEK by setting the parameter `MSK_IPAR_CACHE_LICENSE`.

```
param.MSK_IPAR_CACHE_LICENSE = 'MSK_OFF'; %set parameter.  
[r,res] = mosekopt('minimize',prob,param); %call mosek.
```

It should however be noted that there is a small overhead associated with checking out a license token from the license server.

6.2.1 Waiting for a free license

By default an error will be returned if no license token is available. By setting the parameter **MSK_IPAR_LICENSE_WAIT** MOSEK can be instructed to wait until a license token is available.

```
param.MSK_IPAR_LICENSE_WAIT = 'MSK_ON'; %set parameter.  
[r,res] = mosekopt('minimize',prob,param); %call mosek.
```

Chapter 7

A guided tour

7.1 Introduction

One of the big advantages of MATLAB is that it makes it very easy to do experiments and try out things without doing a lot of programming. The MOSEK optimization toolbox has been designed with this in mind. Hence, it should be very easy to solve optimization problems using MOSEK.

Moreover, a guided tour to the optimization toolbox has been designed to introduce the toolbox by examples. After having studied these examples, the reader should be able to solve his or her own optimization problems without much further effort. Nevertheless, for the user interested in exploiting the toolbox to the limits, a detailed discussion and command reference are provided in the following chapters.

7.2 The tour starts

The MOSEK optimization toolbox consists of two layers of functions. The procedures in the top layer are application specific functions which have an easy-to-use interface. Currently, there are five procedures in the top layer:

<code>msklpopt</code>	Performs linear optimization.
<code>mskqpopt</code>	Performs quadratic optimization.
<code>mskenopt</code>	Performs entropy optimization.
<code>mskgpopt</code>	Performs geometric optimization (posynomial case).
<code>mskscopt</code>	Performs separable convex optimization.

The bottom layer of the MOSEK optimization toolbox consists of one procedure named `mosekopt`. This procedure provides a very flexible and powerful interface to the MOSEK

optimization package. However, the price for this flexibility is a more complicated calling procedure.

For compatibility with the MATLAB optimization toolbox MOSEK also provides an implementation of `linprog`, `quadprog` and so forth. For details about these functions we refer the reader to Chapter 8.

In the following sections usage of the MOSEK optimization toolbox is demonstrated using examples. Most of these examples are available in

`mosek\6\toolbox\examp\`

7.3 The MOSEK terminology

First, some MOSEK terminology is introduced which will make the following sections easy to understand.

The MOSEK optimization toolbox can solve different classes of optimization problems such as linear, quadratic, conic, and mixed-integer optimization problems. Each of these problems is solved by one of the optimizers in MOSEK. Indeed MOSEK includes the following optimizers:

- Interior-point optimizer.
- Conic interior-point optimizer.
- Primal simplex optimizer.
- Mixed-integer optimizer.

Depending on the optimizer different solution types may be produced, e.g. the interior-point optimizers produce a general interior-point solution whereas the simplex optimizer produces a basic solution.

7.4 Linear optimization

The first example is the linear optimization problem

$$\begin{aligned}
 & \text{minimize} && x_1 + 2x_2 \\
 & \text{subject to} && 4 \leq x_1 + x_3 \leq 6, \\
 & && 1 \leq x_1 + x_2, \\
 & && 0 \leq x_1, x_2, x_3.
 \end{aligned} \tag{7.1}$$

7.4.1 Using msklpopt

A linear optimization problem such as (7.1) can be solved using the `msklpopt` function which is designed for solving the problem

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && l^c \leq Ax \leq u^c, \\ & && l^x \leq x \leq u^x. \end{aligned} \tag{7.2}$$

l^c and u^c are called constraint bounds whereas l^x and u^x are variable bounds.

The first step in solving the example (7.1) is to setup the data for problem (7.2) i.e. the c , A , etc. Afterwards the problem is solved using an appropriate call to `msklpopt`.

```
% lo1.m

c      = [1 2 0]';
a      = [[1 0 1];[1 1 0]];
blc    = [4 1]';
buc    = [6 inf]';
blx    = sparse(3,1);
bux    = [];
[res] = msklpopt(c,a,blc,buc,blx,bux);
sol    = res.sol;

% Interior-point solution.

sol.itr.xx'      % x solution.
sol.itr.sux'     % Dual variables corresponding to buc.
sol.itr.slx'     % Dual variables corresponding to blx.

% Basic solution.

sol.bas.xx'      % x solution in basic solution.
```

Please note that

- Infinite bounds are specified using `-inf` and `inf`. Moreover, the `bux = []` means that all upper bounds u^x are plus infinite.
- The `[res] = msklpopt(c,a,blc,buc)` call implies that the lower and upper bounds on x are minus and plus infinity respectively.
- The lines after the `msklpopt` call can be omitted, but the purpose of those lines is to display different parts of the solutions. The `res.sol` field contains one or more solutions. In this case both the interior-point solution (`sol.itr`) and the basic solution (`sol.bas`) are defined.

7.4.2 Using mosekopt

The `msklpopt` function is in fact just a wrapper around the real optimization routine `mosekopt`. Therefore, an alternative to using the `msklpopt` is to call `mosekopt` directly. In general, the syntax for a `mosekopt` call is

```
[rcode,res] = mosekopt(cmd,prob,param)
```

The arguments `prob` and `param` are optional. The purpose of the arguments are as follows:

<code>cmd</code>	A string telling <code>mosekopt</code> what to do, e.g. 'minimize info' tells <code>mosekopt</code> that the objective should be minimized and information about the optimization should be returned.
<code>prob</code>	A MATLAB structure specifying the problem that should be optimized.
<code>param</code>	A MATLAB structure specifying parameters controlling the behavior of the MOSEK optimizer. However, in general it should not be necessary to change the parameters.

The following MATLAB commands demonstrate how to set up the `prob` structure for the example (7.1) and solve the problem using `mosekopt`:

```
% lo2.m

clear prob;

% Specify the c vector.
prob.c = [ 1 2 0]';

% Specify a in sparse format.
subi    = [1 2 2 1];
subj    = [1 1 2 3];
valij   = [1.0 1.0 1.0 1.0];

prob.a = sparse(subi,subj,valij);

% Specify lower bounds of the constraints.
prob.blc = [4.0 1.0]';

% Specify upper bounds of the constraints.
prob.buc = [6.0 inf]';

% Specify lower bounds of the variables.
prob.blx = sparse(3,1);

% Specify upper bounds of the variables.
prob.bux = []; % There are no bounds.
```

```
% Perform the optimization.
[r,res] = mosekopt('minimize',prob);

% Show the optimal x solution.
res.sol.bas.xx
```

Please note that

- A MATLAB structure named **prob** containing all the relevant problem data is defined.
- All fields of this structure are optional except **prob.a** which is required to be a **sparse** matrix.
- Different parts of the solution can be viewed by inspecting the solution field **res.sol**.

7.5 Convex quadratic optimization

A frequently occurring problem type is the quadratic optimization problem which consists of minimizing a quadratic objective function subject to linear constraints. One example of such a problem is:

$$\begin{aligned} & \text{minimize} && x_1^2 + 0.1x_2^2 + x_3^2 - x_1x_3 - x_2 \\ & \text{subject to} && 1 \leq x_1 + x_2 + x_3 \\ & && x \geq 0. \end{aligned} \tag{7.3}$$

In general, a quadratic optimization problem has the form

$$\begin{aligned} & \text{minimize} && \frac{1}{2}x^T Qx + c^T x \\ & \text{subject to} && l^c \leq Ax, \leq u^c, \\ & && l^x \leq x \leq u^x, \end{aligned} \tag{7.4}$$

which for the example (7.3) implies that

$$Q = \begin{bmatrix} 2 & 0 & -1 \\ 0 & 0.2 & 0 \\ -1 & 0 & 2 \end{bmatrix}, \quad c = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, \tag{7.5}$$

and that

$$l^c = 1, \quad u^c = \infty, \quad l^x = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \text{ and } u^x = \begin{bmatrix} \infty \\ \infty \\ \infty \end{bmatrix}$$

Please note the explicit $\frac{1}{2}$ in the objective function of (7.4) which implies that diagonal elements must be doubled in Q , i.e. $Q_{11} = 2$, whereas the coefficient in (7.3) is 1 in front of x_1^2 .

7.5.1 Two important assumptions

MOSEK assumes that the Q matrix is symmetric, i.e.

$$Q = Q^T$$

and that Q is *positive semi-definite*. A matrix is positive semi-definite if the smallest eigenvalue of the matrix is nonnegative. An alternative statement of the positive semi-definite requirement is

$$x^T Q x \geq 0, \forall x.$$

If Q is not positive semi-definite, then MOSEK will not produce reliable results or work at all.

One way of checking whether Q is positive semi-definite is to check whether all the eigenvalues of Q are nonnegative. The MATLAB command `eig` computes all eigenvalues of a matrix.

7.5.2 Using `mskqpopt`

The subsequent MATLAB statements solve the problem (7.3) using the `mskqpopt` MOSEK function

```
% qo1.m

% Set up Q.
q      = [[2 0 -1];[0 0.2 0];[-1 0 2]];

% Set up the linear part of the problem.
c      = [0 -1 0]';
a      = ones(1,3);
blc    = [1.0];
buc    = [inf];
blx    = sparse(3,1);
bux    = [];

% Optimize the problem.
[res] = mskqpopt(q,c,a,blc,buc,blx,bux);

% Show the primal solution.
res.sol.itr.xx
```

It should be clear that the format for calling `mskqpopt` is very similar to calling `msklpopt` except that the Q matrix is included as the first argument of the call. Similarly, the solution can be inspected by viewing the `res.sol` field.

7.5.3 Using mosekopt

The following sequence of MATLAB commands solves the quadratic optimization example by calling `mosekopt` directly.

```
% qo2.m

clear prob;

% c vector.
prob.c = [0 -1 0]';

% Define the data.

% First the lower triangular part of q in the objective
% is specified in a sparse format. The format is:
%
%   Q(prob.qosubi(t),prob.qosubj(t)) = prob.qoval(t), t=1,...,4

prob.qosubi = [ 1  3  2   3]';
prob.qosubj = [ 1  1  2   3]';
prob.qoval  = [ 2 -1 0.2  2]';

% a, the constraint matrix
subi  = ones(3,1);
subj  = 1:3;
valij = ones(3,1);

prob.a = sparse(subi,subj,valij);

% Lower bounds of constraints.
prob.blc = [1.0]';

% Upper bounds of constraints.
prob.buc = [inf]';

% Lower bounds of variables.
prob.blx = sparse(3,1);

% Upper bounds of variables.
prob.bux = []; % There are no bounds.

[r,res] = mosekopt('minimize',prob);

% Display return code.
fprintf('Return code: %d\n',r);

% Display primal solution for the constraints.
res.sol.itr.xc'

% Display primal solution for the variables.
```

```
res.sol.itr.xx'
```

This sequence of commands looks much like the one that was used to solve the linear optimization example using `mosekopt` except that the definition of the Q matrix in `prob. mosekopt` requires that Q is specified in a sparse format. Indeed the vectors `qosubi`, `qosubj`, and `qoval` are used to specify the coefficients of Q in the objective using the principle

$$Q_{\text{qosubi}(t), \text{qosubj}(t)} = \text{qoval}(t), \quad \text{for } t = 1, \dots, \text{length}(\text{qosubi}).$$

An important observation is that due to Q being symmetric, only the lower triangular part of Q should be specified.

7.6 Conic optimization

One way of generalizing a linear optimization problem is to include a constraint of the form

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

in the problem definition where \mathcal{C} is required to be a *convex cone*. The resulting class of problems is known as *conic optimization*.

MOSEK can solve a subset of all conic problems and subsequently it is demonstrated how to solve this subset using the `mosekopt` toolbox function.

7.6.1 The conic optimization problem

A conic optimization problem has the following form

$$\begin{aligned} & \text{minimize} && c^T x + c^f \\ & \text{subject to} && \begin{array}{lll} l^c & \leq & Ax & \leq & u^c, \\ l^x & \leq & x & \leq & u^x, \\ & & x \in \mathcal{C}, \end{array} \end{aligned} \tag{7.6}$$

where \mathcal{C} must satisfy the following requirements. Let

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

be vectors comprised of parts of the decision variable vector x such that each decision variable is a member of exactly **one** x^t vector, e.g.:

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

Next, define

$$\mathcal{C} := \{x \in \mathbb{R}^n : x^t \in \mathcal{C}_t, t = 1, 2, \dots, k\}$$

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:

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

- Rotated quadratic cone:

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

A variable is by default members of the \mathbb{R} set unless it explicitly belongs to a specific cone.

Although the cones MOSEK can handle give rise to a limited class of conic problems it includes linear, quadratic, quadratically constrained optimization, and other classes of nonlinear convex optimization problems. See Section 10.3 for a discussion.

7.6.2 Solving an example

The problem

$$\begin{aligned} & \text{minimize} && x_5 + x_6 \\ & \text{subject to} && x_1 + x_2 + x_3 + x_4 = 1, \\ & && x_1, x_2, x_3, x_4 \geq 0, \\ & && x_5 \geq \sqrt{x_1^2 + x_3^2}, \\ & && x_6 \geq \sqrt{x_2^2 + x_4^2} \end{aligned} \tag{7.7}$$

is an example of a conic quadratic optimization problem. The problem involves some linear constraints and two quadratic cones. The linear constraints are specified as if the problem was a linear problem whereas the cones are specified using a MATLAB cell array¹ named **cones**. **cones** must contain one cell per cone, where a cell must contain the two fields **type** and **sub**. **type** is used to specify the type of the cone and **sub** is used to specify the member variables of the cone.

The following MATLAB code demonstrates how to solve the example (7.7) using MOSEK.

¹If you are not familiar with the MATLAB cell array, then consult the relevant MATLAB documentation.

```

% cqo1.m

clear prob;

% Specify the non-confic part of the problem.

prob.c    = [0 0 0 0 1 1];
prob.a    = sparse([1 1 1 1 0 0]);
prob.blc  = 1;
prob.buc  = 1;
prob.blx  = [0 0 0 0 -inf -inf];
prob.bux  = inf*ones(6,1);

% Specify the cones.
% Define an empty cell array names 'cones' containing one cell per cone.

prob.cones = cell(2,1);

% The first cone is specified.

prob.cones{1}.type = 'MSK_CT_QUAD';
prob.cones{1}.sub  = [5 3 1];

% The second cone is specified.

prob.cones{2}.type = 'MSK_CT_QUAD';
prob.cones{2}.sub  = [6 2 4];

% The field 'type' specifies the cone type, i.e. whether it is quadratic cone
% or rotated quadratic cone. The keys for the two cone types are MSK_CT_QUAD
% MSK_CT_RQUAD respectively.
%
% The field 'sub' specifies the members of the cone, i.e. the above definitions
% imply that  $x(5) \geq \sqrt{x(3)^2 + x(1)^2}$  and  $x(6) * x(2) \geq x(4)^2$ .

% Optimize the problem.

[r,res]=mosekopt('minimize',prob);

% Display the primal solution.

res.sol.itr.xx'

```

Note in particular that:

- No variable can be member of more than one cone. This is not serious restriction — see the following section.
- The \mathbb{R} set is not specified explicitly.

7.6.3 Quadratic and conic optimization

The example

$$\begin{aligned} & \text{minimize} && x_1 + x_2 + x_3 \\ & \text{subject to} && x_1^2 + x_2^2 + x_3^2 \leq 1, \\ & && x_1 + 0.5x_2^2 + x_3 \leq 0.5 \end{aligned} \tag{7.8}$$

is not a conic quadratic optimization problem but can easily be reformulated as such.

Indeed the first constraint is equivalent to

$$\begin{aligned} x_4 &\geq \sqrt{x_1^2 + x_2^2 + x_3^2}, \\ x_4 &= 1 \end{aligned} \tag{7.9}$$

where x_4 is a new variable. This is a quadratic cone and a linear constraint. The second constraint in (7.8) is equivalent to

$$\begin{aligned} x_1 + x_3 + x_5 &= 0.5, \\ x_2 - x_7 &= 0, \\ x_5 &\geq 0, \\ x_6 &= 1, \\ x_7^2 &\leq 2x_5x_6, \end{aligned}$$

because this implies that

$$x_5 \geq 0.5x_7^2 = 0.5x_2^2.$$

and that

$$x_1 + 0.5x_2^2 + x_3 \leq x_1 + x_3 + x_5 = 0.5.$$

Please note that no variable can occur in more than one cone and therefore the additional constraint

$$x_2 = x_7$$

is introduced and x_7 is included in the second conic constraint instead of x_2 . Using this “trick” it is always possible to obtain a formulation where no variable occurs in more than one cone.

Therefore, the example (7.8) is equivalent to the conic quadratic optimization problem

$$\begin{aligned} & \text{minimize} && x_1 + x_2 + x_3 \\ & \text{subject to} && x_1 + x_3 + x_5 = 0.5, \\ & && x_2 - x_7 = 0, \\ & && x_4 = 1, \\ & && x_5 \geq 0, \\ & && x_6 = 1, \\ & && x_4 \geq \sqrt{x_1^2 + x_2^2 + x_3^2}, \\ & && 2x_5x_6 \geq x_7^2. \end{aligned} \tag{7.10}$$

This problem can be solved using MOSEK as follows:

```

% cqp2.m

% Set up the non-conic part of the problem.

prob          = [];
prob.c        = [1 1 1 0 0 0 0]';
prob.a        = sparse([[1 0 1 0 1 0 0];...
                        [0 1 0 0 0 0 -1]]);
prob.blc      = [0.5 0];
prob.buc      = [0.5 0];
prob.blx      = [-inf -inf -inf 1 -inf 1 -inf];
prob.bux      = [inf  inf  inf 1  inf 1  inf];

% Set up the cone information.

prob.cones     = cell(2,1);
prob.cones{1}.type = 'MSK_CT_QUAD';
prob.cones{1}.sub  = [4 1 2 3];
prob.cones{2}.type = 'MSK_CT_RQUAD';
prob.cones{2}.sub  = [5 6 7];

[r,res]        = mosekopt('minimize',prob);

% Display the solution.
res.sol.itr.xx'

```

7.6.4 Conic duality and the dual solution

The dual problem corresponding to the conic optimization problem (7.6) is given by

$$\begin{aligned}
& \text{maximize} && (l^c)^T s_l^c - (u^c)^T s_u^c \\
& && + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f \\
& \text{subject to} && -y + s_l^c - s_u^c = 0, \\
& && A^T y + s_l^x - s_u^x + s_n^x = c, \\
& && s_l^c, s_u^c, s_l^x, s_u^x \geq 0, \\
& && s_n^x \in \mathcal{C}^*
\end{aligned} \tag{7.11}$$

where the dual cone \mathcal{C}^* is defined as follows. Let (s_n^x) be partitioned similar to x , i.e. if x_j is a member of x^t , then $(s_n^x)_j$ is a member of $(s_n^x)^t$ as well. Now, the dual cone is defined by

$$\mathcal{C}^* := \left\{ s_n^x \in \mathbb{R}^{n^t} : (s_n^x)^t \in \mathcal{C}_t^*, t = 1, \dots, k \right\}$$

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

- \mathbb{R} set:

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

- Quadratic cone:

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

- Rotated quadratic cone:

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

For a more detailed discussion about conic duality see Section [10.3](#).

7.6.4.1 How to obtain the dual solution

When solving a conic optimization problem using MOSEK, the dual solution is available. The following MATLAB code fragment shows where the dual solution is stored.

```
% cqo3.m

[r,res]=mosekopt('minimize',prob);

% Solution record.
res.sol

% Dual variables for lower
% bounds of constraints.
res.sol.itr.slc'

% Dual variables for upper
% bounds of constraints.
res.sol.itr.suc'

% Dual variables for lower
% bounds on variables.
res.sol.itr.slx'

% Dual variables for upper
% bounds on variables.
res.sol.itr.sux'

% Dual variables with respect
% to the conic constraints.
res.sol.itr.snx'
```

7.6.5 Setting accuracy parameters for the conic optimizer

Three parameters control the accuracy of the solution obtained by the conic interior-point optimizer. The following example demonstrates which parameters should be reduced to obtain a more accurate solution, if required.

```
% How to change the parameters that controls
% the accuracy of a solution computed by the conic
% optimizer.

param = [];

% Primal feasibility tolerance for the primal solution
param.MSK_DPAR_INTPNT_CO_TOL_PFEAS = 1.0e-8;

% Dual feasibility tolerance for the dual solution
param.MSK_DPAR_INTPNT_CO_TOL_DFEAS = 1.0e-8;

% Relative primal-dual gap tolerance.
param.MSK_DPAR_INTPNT_CO_TOL_REL_GAP = 1.0e-8;

[r,res]=mosekopt('minimize',prob,param);
```

7.7 Quadratically constrained optimization

In the previous section a quadratically constrained optimization problem was solved using the conic optimizer. It is also possible to solve such a problem directly. An example of such an optimization problem is

$$\begin{aligned} \text{minimize} \quad & x_1 + x_2 + x_3 \\ \text{subject to} \quad & x_1^2 + x_2^2 + x_3^2 \leq 1, \\ & x_1 + 0.1x_2^2 + x_3 \leq 0.5. \end{aligned} \tag{7.12}$$

Please note that there are quadratic terms in both constraints. This problem can be solved using `mosekopt` as follows:

```
% qco1.m

clear prob;

% Specify the linear objective terms.
prob.c = ones(3,1);

% Specify the quadratic objective terms.
prob.qcsubk = [1 1 1 2]';
prob.qcsubi = [1 2 3 2]';
prob.qcsubj = [1 2 3 2]';
```

```

prob.qcval = [2.0 2.0 2.0 0.2]';

% Specify the linear constraint matrix
prob.a = [sparse(1,3);sparse([1 0 1])];

prob.buc = [1 0.5]';

[r,res] = mosekopt('minimize',prob);

% Display the solution.
fprintf('\nx:');
fprintf(' %-4e',res.sol.itr.xx');
fprintf('\n||x||: %-4e',norm(res.sol.itr.xx));

```

Note that the quadratic terms in the constraints are specified using the fields `prob.qcsubk`, `prob.qcsubi`, `prob.qcsubj`, and `prob.qcval` as follows

$$Q_{qcsubi(t),qcsubj(t)}^{qcsubk(t)} = qcval(t), \quad \text{for } t = 1, \dots, \text{length}(qcsubk)$$

where $\frac{1}{2}x^T Q^k x$ is the quadratic term in the k th constraint. Also note that only the lower triangular part of the Q^k s should be specified.

7.8 Linear least squares and related norm minimization problems

A frequently occurring problem in statistics and in many other areas of science is the problem

$$\text{minimize} \quad \|Fx - b\| \tag{7.13}$$

where F and b are a matrix and vector of appropriate dimensions. x is the vector decision variables.

Typically, the norm used is the 1-norm, the 2-norm, or the infinity norm.

7.8.1 The case of the 2 norm

Initially let us focus on the 2 norm. In this case (7.13) is identical to the quadratic optimization problem

$$\text{minimize} \quad 1/2x^T F^T Fx + 1/2b^T b - b^T Fx \tag{7.14}$$

in the sense that the set of optimal solutions for the two problems coincides. This fact follows from

$$\begin{aligned} \|Fx - b\|^2 &= (Fx - b)^T (Fx - b) \\ &= x^T F^T Fx + b^T b - 2b^T Fx. \end{aligned}$$

Subsequently, it is demonstrated how the quadratic optimization problem (7.14) is solved using `mosekopt`. In the example the problem data is read from a file, then data for the problem (7.14) is constructed and finally the problem is solved.

```
% nrm1.m

% Read data from 'afiro.mps'.
[r,res] = mosekopt('read(afiro.mps)');

% Get data for the problem
%           minimize ||f x - b||_2
f = res.prob.a';
b = res.prob.c;

% Solve the problem
%           minimize 0.5 xf'fx+0.5*b'*b-(f'*b)'*x

% Clear prob
clear prob;

% Compute the fixed term in the objective.
prob.cfix = 0.5*b'*b

% Create the linear objective terms
prob.c = -f'*b;

% Create the quadratic terms. Please note that only the lower triangular
% part of f'*f is used.
[prob.qosubi,prob.qosubj,prob.qoval] = find(sparse(tril(f'*f)))

% Obtain the matrix dimensions.
[m,n] = size(f);

% Specify a.
prob.a = sparse(0,n);

[r,res] = mosekopt('minimize',prob);

% The optimality conditions are f'*(f x - b) = 0.
% Check if they are satisfied:

fprintf('\nnorm(f^T(fx-b)): %e',norm(f'*(f*res.sol.itr.xx-b)));
```

Often the x variables must be within some bounds or satisfy some additional linear constraints. These requirements can easily be incorporated into the problem (7.14). E.g. the constraint $\|x\|_\infty \leq 1$ can be modeled as follows

```
% nrm2.m. Continuation of nrm1.m.

% Assume that the same objective should be
```

```
% minimized subject to -1 <= x <= 1

prob.blx = -ones(n,1);
prob.bux = ones(n,1);

[r,res] = mosekopt('minimize',prob);

% Check if the solution is feasible.
norm(res.sol.itr.xx,inf)
```

7.8.2 The case of the infinity norm

In some applications of the norm minimization problem (7.13) it is better to use the infinity norm than the 2 norm. However, the problem (7.13) stated as an infinity norm problem is equivalent to the linear optimization problem

$$\begin{aligned} & \text{minimize} && \tau \\ & \text{subject to} && Fx + \tau e - b \geq 0, \\ & && Fx - \tau e - b \leq 0, \end{aligned} \tag{7.15}$$

where e is the vector of ones of appropriate dimension. This implies that

$$\begin{aligned} \tau e &\geq Fx - b \\ \tau e &\geq -(Fx - b) \end{aligned}$$

and hence at optimum

$$\tau^* = \|Fx^* - b\|_\infty$$

holds.

The problem (7.15) is straightforward to solve.

```
% nrm3.m. Continuation of nrm1.m.

% Let x(n+1) play the role as tau, then the problem is
% solved as follows.

clear prob;

prob.c = sparse(n+1,1,1.0,n+1,1);
prob.a = [[f,ones(m,1)],[f,-ones(m,1)]];
prob.blc = [b; -inf*ones(m,1)];
prob.buc = [inf*ones(m,1); b];

[r,res] = mosekopt('minimize',prob);

% The optimal objective value is given by:
norm(f*res.sol.itr.xx(1:n)-b,inf)
```

7.8.3 The case of the 1-norm

By definition, for the 1-norm we have that

$$\|Fx - b\|_1 = \sum_{i=1}^m |f_{i:}x - b_i|.$$

Therefore, the norm minimization problem can be formulated as follows

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^m t_i \\ & \text{subject to} && |f_{i:}x - b_i| = t_i, \quad i = 1, \dots, m, \end{aligned} \tag{7.16}$$

which in turn is equivalent to

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^m t_i \\ & \text{subject to} && \begin{aligned} f_{i:}x - b_i &\leq t_i, & i = 1, \dots, m, \\ -(f_{i:}x - b_i) &\leq t_i, & i = 1, \dots, m. \end{aligned} \end{aligned} \tag{7.17}$$

The reader should verify that this is really the case.

In matrix notation this problem can be expressed as follows

$$\begin{aligned} & \text{minimize} && e^T t \\ & \text{subject to} && \begin{aligned} Fx - te &\leq b, \\ Fx + te &\geq b, \end{aligned} \end{aligned} \tag{7.18}$$

where $e = (1, \dots, 1)^T$. Next, this problem is solved.

```
% nrm4.m. Continuation of nrm1.m.

% Let x(n:(m+n)) play the role as t. Now,
% the problem can be solved as follows

clear prob;

prob.c = [sparse(n,1) ; ones(m,1)];
prob.a = [[f,-speye(m)] ; [f,speye(m)]];
prob.blc = [-inf*ones(m,1); b];
prob.buc = [b ; inf*ones(m,1)];

[r,res] = mosekopt('minimize',prob);

% The optimal objective value is given by:
norm(f*res.sol.itr.xx(1:n)-b,1)
```

7.8.3.1 A better formulation

It is possible to improve upon the formulation of the problem (7.17). Indeed problem (7.17) is equivalent to

$$\begin{aligned}
 & \text{minimize} && \sum_{i=1}^m t_i \\
 & \text{subject to} && f_{i:}x - b_i - t_i + v_i = 0, \quad i = 1, \dots, m, \\
 & && -(f_{i:}x - b_i) - t_i \leq 0, \quad i = 1, \dots, m, \\
 & && v_i \geq 0, \quad i = 1, \dots, m.
 \end{aligned} \tag{7.19}$$

After eliminating the t variables then this problem is equivalent to

$$\begin{aligned}
 & \text{minimize} && \sum_{i=1}^m (f_{i:}x - b_i + v_i) \\
 & \text{subject to} && -2(f_{i:}x - b_i) - v_i \leq 0, \quad i = 1, \dots, m, \\
 & && v_i \geq 0, \quad i = 1, \dots, m.
 \end{aligned} \tag{7.20}$$

Please note that this problem has only half the number of general constraints than problem (7.17) since we have replaced constraints of the general form

$$f_{i:}x \leq b_i$$

with simpler constraints

$$v_i \geq 0$$

which MOSEK treats in a special and highly efficient way. Furthermore MOSEK stores only the non-zeros in the coefficient matrix of the constraints. This implies that the problem (7.20) is likely to require much less space than the problem (7.19).

It is left as an exercise for the reader to implement this formulation in MATLAB.

7.9 More about solving linear least squares problems

Linear least squares problems with and without linear side constraints appear very frequently in practice and it is therefore important to know how such problems are solved efficiently using MOSEK.

Now, assume that the problem of interest is the linear least squares problem

$$\begin{aligned}
 & \text{minimize} && \frac{1}{2} \|Fx - f\|_2^2 \\
 & \text{subject to} && Ax = b, \\
 & && l^x \leq x \leq u^x,
 \end{aligned} \tag{7.21}$$

where F and A are matrices and the remaining quantities are vectors. x is the vector of decision variables. The problem (7.21) as stated is a convex quadratic optimization problem and can be solved as such.

However, if F has much fewer rows than columns then it will usually be more efficient to solve the equivalent problem

$$\begin{aligned} & \text{minimize} && \frac{1}{2} \|z\|_2^2 \\ & \text{subject to} && Ax = b, \\ & && Fx - z = f, \\ & && l^x \leq x \leq u^x. \end{aligned} \tag{7.22}$$

Please note that a number of new constraints and variables has been introduced which of course seems to be disadvantageous but on the other hand the Hessian of the objective in problem (7.22) is much sparser than in problem (7.21). Frequently this turns out to be more important for the computational efficiency and therefore the latter formulation is usually the better one.

If F has many more rows than columns, then formulation (7.22) is not attractive whereas the corresponding dual problem is. Using the duality theory outlined in Section 10.4.1 we obtain the dual problem

$$\begin{aligned} & \text{maximize} && b^T y + f^T \bar{y} \\ & && + (l^x)^T s_l^x + (u^x)^T s_u^x \\ & && - \frac{1}{2} \|z\|_2^2 \\ & \text{subject to} && A^T y + F^T \bar{y} + s_l^x - s_u^x = 0, \\ & && z - \bar{y} = 0, \\ & && s_l^x, s_u^x \geq 0 \end{aligned} \tag{7.23}$$

which can be simplified to

$$\begin{aligned} & \text{maximize} && b^T y + f^T z \\ & && + (l^x)^T s_l^x + (u^x)^T s_u^x \\ & && - \frac{1}{2} \|z\|_2^2 \\ & \text{subject to} && A^T y + F^T z + s_l^x - s_u^x = 0, \\ & && s_l^x, s_u^x \geq 0 \end{aligned} \tag{7.24}$$

after eliminating the \bar{y} variables. Here we use the convention that

$$l_j^x = -\infty \Rightarrow (s_l^x)_j = 0 \quad \text{and} \quad u_j^x = \infty \Rightarrow (s_u^x)_j = 0.$$

In practice such fixed variables in s_l^x and s_u^x should be removed from the problem.

Given our assumptions the dual problem (7.24) will have much fewer constraints than the primal problem (7.22); in general, the fewer constraints a problem contains, the more efficient MOSEK tends to be. A question is: If the dual problem (7.24) is solved instead of the primal problem (7.22), how is the optimal x solution obtained? It turns out that the dual variables corresponding to the constraint

$$A^T y + F^T z + s_l^x - s_u^x = 0$$

are the optimal x solution. Therefore, due to the fact that MOSEK always reports this information as the

```
res.sol.itr.y
```

vector, the optimal x solution can easily be obtained.

In the following code fragment it is investigated whether it is attractive to solve the dual rather than the primal problem for a concrete numerical example. This example has no linear equalities and F is a 2000 by 400 matrix.

```
% nrm5.m

% Read data from a file.
[rcode,res] = mosekopt('read(lsqpd.mps) echo(0)');

% Define the problem data.
F          = res.prob.a;
f          = res.prob.blc;
blx        = res.prob.blx;
bux        = [];

% In this case there are no linear constraints
% First we solve the primal problem:
%
% minimize    0.5|| z ||^2
% subject to  F x - z = f
%              1 <= x <= u

% Note that m>>n
[m,n]       = size(F);

prob        = [];

prob.qosubi = n+(1:m);
prob.qosubj = n+(1:m);
prob.qoval  = ones(m,1);
prob.a      = [F,-speye(m,m)];
prob.blc    = f;
prob.buc    = f;
prob.blx    = [blx;-inf*ones(m,1)];
prob.bux    = bux;

fprintf('m=%d  n=%d\n',m,n);

fprintf('First try\n');

tic
[rcode,res] = mosekopt('minimize echo(0)',prob);
```

```

% Display the solution time.
fprintf('Time           : %-.2f\n',toc);

try
    % x solution:
    x = res.sol.itr.xx;

    % objective value:
    fprintf('Objective value: %-.6e\n',norm(F*x(1:n)-f)^2);

    % Check feasibility.
    fprintf('Feasibility      : %-.6e\n',min(x(1:n)-blx(1:n)));
catch
    fprintf('MSKERROR: Could not get solution')
end

% Clear prob.
prob=[];

%
% Next, we solve the dual problem.

% Index of lower bounds that are finite:
lfin      = find(blx>-inf);

% Index of upper bounds that are finite:
ufin      = find(bux<inf);

prob.qosubi = 1:m;
prob.qosubj = 1:m;
prob.qoval  = -ones(m,1);
prob.c      = [f;blx(lfin);-bux(ufin)];
prob.a      = [F',...
               sparse(lfin,(1:length(lfin))',...
                       ones(length(lfin),1),...
                       n,length(lfin)),...
               sparse(ufin,(1:length(ufin))',...
                       -ones(length(ufin),1),...
                       n,length(ufin))];

prob.blc    = sparse(n,1);
prob.buc    = sparse(n,1);
prob.blx    = [-inf*ones(m,1);...
               sparse(length(lfin)+length(ufin),1)];
prob.bux    = [];

fprintf('\n\nSecond try\n');
tic
[rcode,res] = mosekopt('maximize echo(0)',prob);

% Display the solution time.

```

```

fprintf('Time          : %-.2f\n',toc);

try
    % x solution:
    x = res.sol.itr.y;

    % objective value:
    fprintf('Objective value: %-.6e\n',...
            norm(F*x(1:n)-f)^2);

    % Check feasibility.
    fprintf('Feasibility   : %-.6e\n',...
            min(x(1:n)-blx(1:n)));
catch
    fprintf('MSKERROR: Could not get solution')
end

```

Here is the output produced:

```

m=2000  n=400
First try
Time          : 2.07
Objective value: 2.257945e+001
Feasibility    : 1.466434e-009

```

```

Second try
Time          : 0.47
Objective value: 2.257945e+001
Feasibility    : 2.379134e-009

```

Both formulations produced a strictly feasible solution having the same objective value. Moreover, using the dual formulation leads to a reduction in the solution time by about a factor 5: In this case we can conclude that the dual formulation is far superior to the primal formulation of the problem.

7.9.1 Using conic optimization on linear least squares problems

Linear least squares problems can also be solved using conic optimization because the linear least squares problem

$$\begin{aligned}
 & \text{minimize} && \|Fx - f\|_2 \\
 & \text{subject to} && Ax = b, \\
 & && l^x \leq x \leq u^x
 \end{aligned} \tag{7.25}$$

is equivalent to

$$\begin{aligned}
& \text{minimize} && t \\
& \text{subject to} && Ax = b, \\
& && Fx - z = f, \\
& && l^x \leq x \leq u^x, \\
& && \|z\|_2 \leq t.
\end{aligned} \tag{7.26}$$

This problem is a conic quadratic optimization problem having one quadratic cone and the corresponding dual problem is

$$\begin{aligned}
& \text{maximize} && b^T y + f^T \bar{y} + (l^x)^T s_l^x - (u^x)^T s_u^x \\
& \text{subject to} && A^T y + F^T \bar{y} + s_l^x - s_u^x = 0, \\
& && -\bar{y} + s_z = 0, \\
& && s_t = 1, \\
& && \|s_z\| \leq s_t, \\
& && s_l^x, s_u^x \geq 0
\end{aligned} \tag{7.27}$$

which can be reduced to

$$\begin{aligned}
& \text{maximize} && b^T y + f^T s_z + (l^x)^T s_l^x - (u^x)^T s_u^x \\
& \text{subject to} && A^T y - F^T \bar{s}_z + s_l^x - s_u^x = 0, \\
& && s_t = 1, \\
& && \|s_z\| \leq s_t, \\
& && s_l^x, s_u^x \geq 0.
\end{aligned} \tag{7.28}$$

Often the dual problem has much fewer constraints than the primal problem. In such cases it will be more efficient to solve the dual problem and obtain the primal solution x as the dual solution of the dual problem.

7.10 Entropy optimization

7.10.1 Using `mskenopt`

An entropy optimization problem has the following form

$$\begin{aligned}
& \text{minimize} && \sum_{j=1}^n d_j x_j \ln(x_j) + c^T x \\
& \text{subject to} && l^c \leq Ax \leq u^c, \\
& && 0 \leq x,
\end{aligned} \tag{7.29}$$

where all the components of d must be nonnegative, i.e. $d_j \geq 0$. An example of an entropy optimization problem is

$$\begin{aligned}
& \text{minimize} && x_1 \ln(x_1) - x_1 + x_2 \ln(x_2) \\
& \text{subject to} && 1 \leq x_1 + x_2 \leq 1, \\
& && 0 \leq x_1, x_2.
\end{aligned} \tag{7.30}$$

This problem can be solved using the `mskenopt` command as follows

```
d      = [1 1]';
c      = [-1 0]';
a      = [1 1];
blc    = 1;
buc    = 1;
[res] = mskenopt(d,c,a,blc,buc);
res.sol.itr.xx;
```

7.11 Geometric optimization

A so-called geometric optimization problem can be stated as follows

$$\begin{aligned} & \text{minimize} && \sum_{k \in J_0} c_k \prod_{j=1}^n t_j^{a_{kj}} \\ & \text{subject to} && \sum_{k \in J_i} c_k \prod_{j=1}^n t_j^{a_{kj}} \leq 1, \quad i = 1, \dots, m, \\ & && t > 0, \end{aligned} \tag{7.31}$$

where it is assumed that

$$\cup_{k=0}^m J_k = \{1, \dots, T\}$$

and if $i \neq j$, then

$$J_i \cap J_j = \emptyset.$$

Hence, A is a $T \times n$ matrix and c is a vector of length t . In general, the problem (7.31) is very hard to solve, but the posynomial case where

$$c > 0$$

is relatively easy. Using the variable transformation

$$t_j = e^{x_j} \tag{7.32}$$

we obtain the problem

$$\begin{aligned} & \text{minimize} && \sum_{k \in J_0} c_k e^{a_{k:}x} \\ & \text{subject to} && \sum_{k \in J_i} c_k e^{a_{k:}x} \leq 1, \quad i = 1, \dots, m, \end{aligned} \tag{7.33}$$

which is convex in x for $c > 0$. We apply the log function to obtain the equivalent problem

$$\begin{aligned} & \text{minimize} && \log\left(\sum_{k \in J_0} c_k e^{a_{k:}x}\right) \\ & \text{subject to} && \log\left(\sum_{k \in J_i} c_k e^{a_{k:}x}\right) \leq \log(1), \quad i = 1, \dots, m, \end{aligned} \tag{7.34}$$

which is also a convex optimization problem since log is strictly increasing. Hence, the problem (7.34) can be solved by MOSEK.

For further details about geometric optimization we refer the reader to [11, pp. 531-538].

7.11.1 Using `mskgpopt`

MOSEK cannot handle a geometric optimization problem directly, but the transformation (7.34) can be solved using the MOSEK optimization toolbox function `mskgpopt`. Please note that the solution to the transformed problem can easily be converted into a solution to the original geometric optimization problem using relation (7.32).

Subsequently, we will use the example

$$\begin{aligned} &\text{minimize} && 40t_1^{-1}t_2^{-1/2}t_3^{-1} + 20t_1t_3 + 40t_1t_2t_3 \\ &\text{subject to} && \frac{1}{3}t_1^{-2}t_2^{-2} + \frac{4}{3}t_2^{1/2}t_3^{-1} \leq 1, \\ &&& 0 < t_1, t_2, t_3 \end{aligned} \quad (7.35)$$

to demonstrate how a geometric optimization problem is solved using `mskgpopt`. Please note that both the objective and the constraint functions consist of a sum of simple terms. These terms can be specified completely using the matrix

$$A = \begin{bmatrix} -1 & -0.5 & -1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ -2 & -2 & 0 \\ 0 & 0.5 & -1 \end{bmatrix},$$

and the vectors

$$c = \begin{bmatrix} 40 \\ 20 \\ 40 \\ \frac{1}{3} \\ \frac{4}{3} \\ \frac{1}{3} \end{bmatrix} \text{ and } \text{map} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

The interpretation is this: Each row of A , c describes one term, e.g. the first row of A and the first element of c describe the first term in the objective function. The vector map indicated whether a term belongs to the objective or to a constraint. If map_k equals zero, the k th term belongs to the objective function, otherwise it belongs to the map_k th constraint.

The following MATLAB code demonstrates how the example is solved using `mskgpopt`.

```
% go1.m

c      = [40 20 40 1/3 4/3]';
a      = sparse([-1 -0.5 -1];[1 0 1];...
               [1 1 1];[-2 -2 0];[0 0.5 -1]);
map     = [0 0 0 1 1]';
[res]   = mskgpopt(c,a,map);

fprintf('\nPrimal optimal solution to original gp:');
fprintf(' %e',exp(res.sol.itr.xx));
fprintf('\n\n');
```

```

% Compute the optimal objective value and
% the constraint activities.
v = c.*exp(a*res.sol.itr.xx);

% Add appropriate terms together.
f = sparse(map+1,1:5,ones(size(map)))*v;

% First objective value. Then constraint values.
fprintf('Objective value: %e\n',log(f(1)));
fprintf('Constraint values:');
fprintf(' %e',log(f(2:end)));
fprintf('\n\n');

% Dual multipliers (should be negative)
fprintf('Dual variables (should be negative):');
fprintf(' %e',res.sol.itr.y);
fprintf('\n\n');

```

The code also computes the objective value and the constraint values at the optimal solution. Moreover, the optimal dual Lagrange multipliers for the constraints are shown and the gradient of the Lagrange function at the optimal point is computed.

7.11.2 Comments

7.11.2.1 Solving large scale problems

If you want to solve a large problem, i.e. a problem where A has large dimensions, then A must be sparse or you will run out of space. Recall that a sparse matrix contains few non-zero elements, so if A is a sparse matrix, you should construct it using MATLAB's `sparse` as follows

```
A = sparse(subi,subj,valid);
```

where

$$a_{\text{subi}[k],\text{subj}[k]} = \text{valid}[k].$$

For further details on the `sparse` function, please enter

```
help sparse
```

in MATLAB.

7.11.2.2 Preprocessing tip

Before solving a geometric optimization problem it is worthwhile to check if a column of the A matrix inputted to `mskgpopt` contains only positive elements. If this is the case, the corresponding variable t_i can take the value zero in the optimal solution: This may cause

problems for MOSEK so it is better to remove such variables from the problem — doing so will have no influence on the optimal solution.

7.12 Separable convex optimization

This section discusses separable convex nonlinear optimization problems. A general separable nonlinear optimization problem can be specified as follows:

$$\begin{aligned}
 & \text{minimize} && f(x) + c^T x \\
 & \text{subject to} && g(x) + Ax - x^c = 0, \\
 & && l^c \leq x^c \leq u^c, \\
 & && l^x \leq x \leq u^x,
 \end{aligned} \tag{7.36}$$

where

- m is the number of constraints.
- n is the number of decision variables.
- $x \in \mathbb{R}^n$ is a vector of decision variables.
- $x^c \in \mathbb{R}^m$ is a vector of slack 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.
- $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a nonlinear function.
- $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a nonlinear vector function.

This implies that the i th constraint essentially has the form

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

when the x_i^c variable has been eliminated.

The problem (7.36) must satisfy the three important requirements:

1. Separability: This requirement implies that all nonlinear functions can be written on the form

$$f(x) = \sum_{j=1}^n f^j(x_j)$$

and

$$g_i(x) = \sum_{j=1}^n g_i^j(x_j).$$

Hence, the nonlinear functions can be written as a sum of functions which depends on only one variable.

2. Differentiability: All functions should be twice differentiable for all x_j satisfying

$$l_j^x < x < u_j^x$$

if x_j occurs in at least one nonlinear function. Hence, if $\sqrt{x_2}$ appears in the problem, then the lower bound on x_2 should be 0.

3. Convexity: The problem should be a convex optimization problem. See Section 10.4 for a discussion of this requirement.

7.12.1 Using `mskscopt`

Subsequently, we will use the following example

$$\begin{aligned} &\text{minimize} && x_1 - \ln(x_1 + 2x_2) \\ &\text{subject to} && x_1^2 + x_2^2 \leq 1 \end{aligned} \tag{7.37}$$

to demonstrate solving a convex separable optimization problem using the MOSEK optimization toolbox function `mskscopt`.

First, note that the problem (7.37) is not a separable optimization problem due to the fact that the logarithmic term in objective is not a function of a single variable. However, by introducing one additional constraint and variable the problem can be made separable as follows

$$\begin{aligned} &\text{minimize} && x_1 - \ln(x_3) \\ &\text{subject to} && x_1^2 + x_2^2 \leq 1, \\ & && x_1 + 2x_2 - x_3 = 0, \\ & && x_3 \geq 0. \end{aligned} \tag{7.38}$$

This problem is separable and equivalent to the previous problem. Moreover, note that all nonlinear functions are well defined for values of x satisfying the variable bounds strictly, i.e.

$$x_3 > 0.$$

This (almost) makes sure that function evaluation errors will not occur during the optimization process since MOSEK will only evaluate $\ln(x_3)$ for $x_3 > 0$.

When using the `mskscopt` function to solve problem (7.38), the linear part of the problem, such as a c and A , is specified as usual using MATLAB vectors and matrices. However, the nonlinear functions must be specified using five arrays which in the case of problem (7.38) can have the form

```
opr = ['log'; 'pow'; 'pow'];
opri = [0; 1; 1];
oprj = [3; 1; 2];
oprj = [-1; 1; 1];
oprj = [0; 2; 2];
```

Hence, `opr(k,:)` specifies the type of a nonlinear function, `opri(k)` specifies in which constraint the nonlinear function should be added (zero means objective), and `oprj(k)` means that the nonlinear function should be applied to x_j . Finally, `oprj(k)` and `oprj(k)` are parameters used by the `mskscopt` function according to the table:

<code>opr(k,:)</code>	<code>opri(k)</code>	<code>oprj(k)</code>	<code>oprj(k)</code>	<code>oprj(k)</code>	function
ent	i	j	f	(not used)	$f x_j \ln(x_j)$
exp	i	j	f	g	$f e^{g x_j}$
log	i	j	f	(not used)	$f \ln(x_j)$
pow	i	j	f	g	$f x_j^g$

The i value indicates which constraint the nonlinear function belongs to. However, if i is identical to zero, then the function belongs to the objective. Using this notation a separable convex optimization problem can be solved with the function:

```
mskscopt(opr,
         opri,
         oprj,
         oprf,
         oprg,
         c,
         a,
         blc,
         buc,
         blx,
         bux)
```

All the elements for solving a separable convex nonlinear optimization problem have now been discussed and therefore we will conclude this section by showing the MATLAB code that will solve the example problem (7.38).

```

% sco1.m

% Specify the linear part of the problem.

c          = [1;0;0];
a          = sparse([0 0 0];[1 2 -1]);
blc        = [-inf; 0];
buc        = [1;0];
blx        = [-inf;-inf;0];

% Specify the nonlinear part.

opr        = ['log'; 'pow'; 'pow'];
opri       = [0;      1;      1      ];
oprj       = [3;      1;      2      ];
oprj       = [-1;     1;      1      ];
oprj       = [0;      2;      2      ];

% Call the optimizer.
% Note that bux is an optional parameter which should be added if the variables
% have an upper bound.

[res]      = msksopt(opr,opri,oprj,oprj,oprj,c,a,blc,buc,blx);

% Print the solution.
res.sol.itr.xx

```

7.13 Mixed-integer optimization

Up until now it has been assumed that the variables in an optimization problem are continuous. Hence, it has been assumed that any value between the bounds on a variable is feasible. In many cases this is not a valid assumption because some variables are integer-constrained. E.g. a variable may denote the number of persons assigned to a given job and it may not be possible to assign a fractional person.

Using a mixed-integer optimizer MOSEK is capable of solving linear and quadratic optimization problems where one or more of the variables are integer-constrained.²

²The mixed-integer optimizer is a separately licensed option.

7.13.1 Solving an example

Using the example

$$\begin{aligned}
 &\text{minimize} && -2x_1 - 3x_2 \\
 &\text{subject to} && 195x_1 + 273x_2 \leq 1365, \\
 & && 4x_1 + 40x_2 \leq 140, \\
 & && x_1 \leq 4, \\
 & && x_1, x_2 \geq 0, \quad \text{and integer}
 \end{aligned} \tag{7.39}$$

we will demonstrate how to solve an integer optimization problem using MOSEK.

```

% milo1.m

% Specify the linear problem data as if
% the problem is a linear optimization
% problem.

clear prob
prob.c      = [-2 -3];
prob.a      = sparse([[195 273];[4 40]]);
prob.blc    = -[inf inf];
prob.buc    = [1365 140];
prob.blx    = [0 0];
prob.bux    = [4 inf];

% Specify indexes of variables that are integer
% constrained.

prob.ints.sub = [1 2];

% Optimize the problem.
[r,res] = mosekopt('minimize',prob);

try
    % Display the optimal solution.
    res.sol.int
    res.sol.int.xx'
catch
    fprintf('MSKERROR: Could not get solution')
end

```

Please note that compared to a linear optimization problem with no integer-constrained variables:

- The `prob.ints.sub` field is used to specify the indexes of the variables that are integer-constrained.
- The optimal integer solution is returned in the `res.sol.int` MATLAB structure.

7.13.2 Speeding up the solution of a mixed-integer problem

In general, a mixed-integer optimization problem can be very difficult to solve. Therefore, in some cases it may be necessary to improve upon the problem formulation and “assist” the mixed-integer optimizer.

How to obtain a good problem formulation is beyond the scope of this section and the reader is referred to [26]. However, two methods for assisting the mixed-integer optimizer are discussed subsequently.

7.13.2.1 Specifying an initial feasible solution

In many cases a good feasible integer solution to the optimization problem may be known. If this is the case, it is worthwhile to inform the mixed-integer optimizer since this will reduce the solution space searched by the optimizer.

Consider the problem:

$$\begin{aligned} & \text{maximize} && 7x_0 + 10x_1 + x_2 + 5x_3 \\ & \text{subject to} && x_0 + x_1 + x_2 + x_3 \leq 2.5 \\ & && x_3 \geq 0 \\ & && x_0, x_1, x_2 \geq 0 \quad \text{and integer,} \end{aligned} \tag{7.40}$$

where only some of the variables are integer and the remaining are continuous. A feasible solution to this problem is:

$$x_0 = 0, x_1 = 2, x_2 = 0, x_3 = 0.5 \tag{7.41}$$

The following example demonstrates how to input this initial solution to MOSEK.

```
% milo2.m

clear prob
clear param
[r,res] = mosekopt('symbcon');
sc = res.symbcon;

prob.c = [7 10 1 5];
prob.a = sparse([1 1 1 1]);
prob.blc = -[inf];
prob.buc = [2.5];
prob.blx = [0 0 0 0];
prob.bux = [inf inf inf inf];
prob.ints.sub = [1 2 3];

prob.sol.int.xx = [0 2 0 0.5]';

% Optionally set status keys too.
```

```
% prob.sol.int.skx = [sc.MSK_SK_SUPBAS;sc.MSK_SK_SUPBAS;...
%                  sc.MSK_SK_SUPBAS;sc.MSK_SK_BAS]
% prob.sol.int.skx = [sc.MSK_SK_UPR]

[r,res] = mosekopt('maximize',prob);

try
    % Display the optimal solution.
    res.sol.int.xx'
catch
    fprintf('MSKERROR: Could not get solution')
end
```

It is also possible to specify only the values of the integer variables and then let MOSEK compute values for the remaining continuous variables in order to obtain a feasible solution. If the `MSK_IPAR_MIO_CONSTRUCT_SOL` parameter is set to `MSK_ON` then MOSEK tries to compute a feasible solution from the specified values of the integer variables. MOSEK generates the feasible solution by temporarily fixing all integer variables to the specified values and then optimizing the resulting continuous linear optimization problem. Hence, using this feature it is necessary to specify only the values of `prob.sol.int.xx` corresponding to the integer-constrained variables.

Suppose it is known that $x_0 = 0, x_1 = 2, x_2 = 0$ are candidates for good integer values to our problem, then the following example demonstrates how to optimize the problem (7.40) using a feasible starting solution generated from the integer values as $x_0 = 0, x_1 = 2, x_2 = 0$.

```
% milo3.m

[r,res]      = mosekopt('symbcon');
sc           = res.symbcon;

clear prob

prob.c       = [7 10 1 5];
prob.a       = sparse([1 1 1 1]);
prob.blc     = -[inf];
prob.buc     = [2.5];
prob.blx     = [0 0 0 0];
prob.bux     = [inf inf inf inf];
prob.ints.sub = [1 2 3];

% Values for the integer variables are specified.
prob.sol.int.xx = [0 2 0 0]';

% Tell Mosek to construct a feasible solution from a given integer
% value.
param.MSK_IPAR_MIO_CONSTRUCT_SOL = sc.MSK_ON;

[r,res] = mosekopt('maximize',prob,param);
```

```

try
    % Display the optimal solution.
    res.sol.int.xx'
catch
    fprintf('MSKERROR: Could not get solution')
end

```

7.13.2.2 Using branching priorities

The mixed-integer optimizer in MOSEK employs the so-called *branch-and-bound* algorithm to search for the optimal solution. See [26, pp. 91-112] for details about the branch-and-bound algorithm. The branch-and-bound algorithm can benefit from knowing about priorities of the integer variables.

E.g. in an optimization model some integer variables may denote which factories to build and other variables which products to make in the factories. It seems natural to decide upon which factories to build first and then decide upon which products to make in which factories. Hence, some integer variables are more important than others.

In MOSEK it is possible to assign priorities to all the integer variables. The higher priority assigned to a variable the more important the variable is considered by the branch-and-bound algorithm. Priorities are specified using the `prob.ints.pri` field as follows:

```

prob.ints.sub = [4 1 2 3]; % Integer variables.
prob.ints.pri = [5 10 2 4]; % Priorities.

```

This implies that variable 4 has priority 5, variable 1 has priority 10 and so forth.

An example of the usage of priorities can be seen in [26, pp. 232-235].

7.14 Sensitivity analysis

Given an optimization problem it is often useful to obtain information about how the optimal objective value changes when a problem parameter is perturbed. E.g. the objective function may reflect the price of a raw material such as oil which may not be known with certainty. Therefore, it is interesting to know how the optimal objective value changes as the oil price changes.

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

Consider the problem:

minimize

$$1x_{11} + 2x_{12} + 5x_{23} + 2x_{24} + 1x_{31} + 2x_{33} + 1x_{34} \quad (7.42)$$

subject to

$$\begin{array}{rcl}
 x_{11} + x_{12} & & \leq 400, \\
 & x_{23} + x_{24} & \leq 1200, \\
 & & x_{31} + x_{33} + x_{34} \leq 1000, \\
 x_{11} & & + x_{31} = 800, \\
 & x_{12} & = 100, \\
 & & x_{23} + x_{33} = 500, \\
 & & x_{24} + x_{34} = 500, \\
 x_{11}, & x_{12}, & x_{23}, & x_{24}, & x_{31}, & x_{33}, & x_{34} \geq 0.
 \end{array} \tag{7.43}$$

The example below demonstrate how sensitivity analysis can answer questions of the type: What happens to the optimal solution if we decrease the upper bound of the first constraint with 1? For more information on sensitivity analysis see Chapter 14.

```
% sensitivity2.m

% Setup problem data.
clear prob
prob.a = sparse([1, 1, 0, 0, 0, 0, 0;
                0, 0, 1, 1, 0, 0, 0;
                0, 0, 0, 0, 1, 1, 1;
                1, 0, 0, 0, 1, 0, 0;
                0, 1, 0, 0, 0, 0, 0;
                0, 0, 1, 0, 0, 1, 0;
                0, 0, 0, 1, 0, 0, 1]);

prob.c = [1,2,5,2,1,2,1];
prob.blc = [-Inf,-Inf,-Inf,800,100,500, 500];
prob.buc = [400,1200,1000,800,100,500,500];
prob.bux(1:7) = Inf;
prob.blx(1:7) = 0;

% Analyze upper bound of constraint 1.
prob.prisen.cons.subu = [1];

[r,res] = mosekopt('minimize echo(0)',prob);
fprintf('Optimal objective value: %e\n',prob.c * res.sol.bas.xx );
fprintf('Sensitivity results for constraint 1:');
res.prisen.cons

% If we change the upper bound of constraint 1 with a
% value v in [res.prisen.cons.lr_bu(1),res.prisen.cons.rr_bu(1)]
% then the optimal objective changes with - v * ls_bu(0)
% e.g. changing prob.buc(1) with -1
prob.buc(1) = prob.buc(1) - 1;
new_sol_predicted = prob.c * res.sol.bas.xx + 1 * res.prisen.cons.ls_bu(1);
fprintf('New optimal objective after changing bound predicted to:%e\n', ...
        new_sol_predicted);
```

```
[r,res] = mosekopt('minimize echo(0)',prob);
fprintf ('New optimal objective value: %e\n',prob.c * res.sol.bas.xx );
```

The output from running the example is given below:

```
Optimal objective value: 3.000000e+03
Sensitivity results for constraint 1:
ans =
```

```
lr_bl: []
rr_bl: []
ls_bl: []
rs_bl: []
lr_bu: -300
rr_bu: 0
ls_bu: 3
rs_bu: 3
```

```
New optimal objective after changing bound predicted to:3.003000e+03
New optimal objective value: 3.003000e+03
```

7.15 Inspecting a problem

The problem analyzer (discussed in detail in Sec. [13.1](#)) provides useful diagnostics about an optimization problem, and is quick way to verify that a model has been specified correctly. For example, executing the command

```
mosekopt('anapro',prob)
```

will generate a report looking like

Constraints		Bounds	Variables
upper bd:	19	lower bd: all	cont: all
fixed :	8		

```
-----

Objective, cx
  range: min |c|: 0.00000  min |c|>0: 0.320000  max |c|: 10.0000
distrib:      |c|      vars
           0      27
        [0.32, 1)      4
```

[1, 10] 1

Constraint matrix A has

27 rows (constraints)
 32 columns (variables)
 83 (9.60648%) nonzero entries (coefficients)

Row nonzeros, A_i

range: min A_i: 1 (3.125%) max A_i: 9 (28.125%)

distrib:	A _i	rows	rows%	acc%
	1	2	7.41	7.41
	2	16	59.26	66.67
	[3, 7]	8	29.63	96.30
	[8, 9]	1	3.70	100.00

Column nonzeros, A_j

range: min A_j: 1 (3.7037%) max A_j: 4 (14.8148%)

distrib:	A _j	cols	cols%	acc%
	1	1	3.12	3.12
	2	21	65.62	68.75
	[3, 4]	10	31.25	100.00

A nonzeros, A(ij)

range: min |A(ij)|: 0.107000 max |A(ij)|: 2.42900

distrib:	A(ij)	coeffs
	[0.107, 1)	17
	[1, 2.43]	66

Constraint bounds, lb ≤ Ax ≤ ub

distrib:	b	lbs	ubs
	0	7	20
	[10, 100)	1	3
	[100, 1000]		4

Variable bounds, lb ≤ x ≤ ub

distrib:	b	lbs	ubs
	0	32	

The report provides an overview of the objective function, the number of constraints and bounds, as well as sparsity information and distributions of nonzero elements.

7.16 The solutions

Whenever an optimization problem is solved using MOSEK, one or more optimal solutions are reported depending on which optimizer is used. These solutions are available in the

`res.sol`

structure, which has one or more of the subfields

```
res.sol.itr % Interior solution.
res.sol.bas % Basic      solution.
res.sol.int % Integer  solution.
```

The interior (point) solution is an arbitrary optimal solution which is computed using the interior-point optimizer. The basic solution is available only for linear problems and is produced by the simplex optimizer or the basis identification process which is an add-on to the interior-point optimizer. Finally, the integer solution is available only for problems having integer-constrained variables and is computed using the integer optimizer.

Each of the three solutions may contain one or more of the following subfields:

<code>.prosta</code>	Problem status. See Appendix F.38 .
<code>.solsta</code>	Solution status. See Appendix F.51 .
<code>.skc</code>	Constraint status keys. See Table 7.1 below.
<code>.skx</code>	Variable status keys. See Table 7.1 below.
<code>.xc</code>	Constraint activities.
<code>.xx</code>	Variable activities.
<code>.y</code>	Identical to <code>-.slc+.suc</code> .
<code>.slc</code>	Dual variables corresponding to lower constraint bounds.
<code>.suc</code>	Dual variables corresponding to upper constraint bounds.
<code>.slx</code>	Dual variables corresponding to lower variable bounds.
<code>.sux</code>	Dual variables corresponding to upper variable bounds.
<code>.snx</code>	Dual variables corresponding to the conic constraints.

7.16.1 The constraint and variable status keys

In a solution both constraints and variables are assigned a status key which indicates whether the constraint or variable is at its lower limit, its upper limit, is super basic and so forth in the optimal solution. For interior-point solutions these status keys are only indicators which the optimizer produces.

In Table 7.1 the possible values for the status keys are shown accompanied with an interpretation of the key.

Symbolic constant	Numeric constant	String code	Interpretation
MSK_SK_UNK	0	UN	Unknown status
MSK_SK_BAS	1	BS	Is basic
MSK_SK_SUPBAS	2	SB	Is superbasic
MSK_SK_LOW	3	LL	Is at the lower limit (bound)
MSK_SK_UPR	4	UL	Is at the upper limit (bound)
MSK_SK_FIX	5	EQ	Lower limit is identical to upper limit
MSK_SK_INF	6	**	Is infeasible i.e. the lower limit is greater than the upper limit.

Table 7.1: Constraint and variable status keys.

By default the constraint and variable status keys are reported using string codes but it is easy to have MOSEK report the numeric codes instead. Indeed in the example

```
% Status keys in string format.
[rcode,res]=mosekopt('minimize statuskeys(0)',prob);
res.sol.skc(1)
res.sol.prosta
```

the status keys are represented using string codes whereas in the example

```
% Status keys in numeric format.
[rcode,res]=mosekopt('minimize statuskeys(1)',prob);
res.sol.skc(1)
res.sol.prosta
```

the status keys are represented using numeric codes.

7.17 Viewing the task information

In MOSEK the optimization problem and the related instructions with respect to the optimization process are called an optimization task or for short a task. Whenever MOSEK

performs operations on a task it stores information in the task information database. Examples of information that is stored are the number of interior-point iterations performed to solve the problem and time spent doing the optimization.

All the items stored in the task information database are listed in Appendixes [F.13](#) and [F.17](#). It is possible to see the whole or part of the task information database from within MATLAB.

```
% Solve a problem and obtain
% the task information database.
[r,res]=mosekopt('minimize info',prob);

% View one item
res.info.MSK_IINF_INTPNT_ITER

% View the whole database
res.info
```

7.18 Inspecting and setting parameters

A large number of parameters controls the behavior of MOSEK, e.g. there is a parameter controlling which optimizer is used, one that limits the maximum number of iterations allowed, and several parameters specifying the termination tolerance. All these parameters are stored in a database internally in MOSEK. The complete parameter database can be obtained and viewed using the commands:

```
[r,res]=mosekopt('param');
res.param
```

We will not describe the purpose of each parameter here but instead refer the reader to Appendix [E](#) where all the parameters are presented in detail.

In general, it should not be necessary to change any of the parameters but if required, it is easily done. In the following example code it is demonstrated how to modify a few parameters and afterwards performing the optimization using these parameters.

```
% Obtain all symbolic constants
% defined by MOSEK.

[r,res] = mosekopt('symbcon');
sc      = res.symbcon;

param   = [];

% Basis identification is unnecessary.
param.MSK_IPAR_INTPNT_BASIS = sc.MSK_OFF;

% Alternatively you can use
```

```
%
% param.MSK_IPAR_INTPNT_BASIS    = 'MSK_OFF';
%

% Use another termination tolerance.
param.MSK_DPAR_INTPNT_TOLRGAP = 1.0e-9;

% Perform optimization using the
% modified parameters.

[r,res] = mosekopt('minimize',prob,param);
```

7.19 Advanced start (hot-start)

In practice it frequently occurs that when an optimization problem has been solved, then the same problem slightly modified should be reoptimized. Moreover, if it is just a small the modification, it can be expected that the optimal solution to the original problem is a good approximation to the modified problem. Therefore, it should be efficient to start the optimization of the modified problem from the previous optimal solution.

Currently, the interior-point optimizer in MOSEK **cannot** take advantage of a previous optimal solution, however, the simplex optimizer can exploit any basic solution.

7.19.1 Some examples using hot-start

Using the example

$$\begin{aligned}
 &\text{minimize} && x_1 + 2x_2 \\
 &\text{subject to} && 4 \leq x_1 + x_3 \leq 6, \\
 & && 1 \leq x_1 + x_2, \\
 & && 0 \leq x_1, x_2, x_3
 \end{aligned} \tag{7.44}$$

the hot-start facility using the simplex optimizer will be demonstrated. A quick inspection of the problem indicates that $(x_1, x_3) = (1, 3)$ is an optimal solution. Hence, it seems to be a good idea to let the initial basis consist of x_1 and x_3 and all the other variables be at their lower bounds. This idea is used in the example code:

```
% advs1.m

clear prob param bas

% Specify an initial basic solution.
bas.skc    = ['LL','LL'];
bas.skx    = ['BS','LL','BS'];
bas.xc     = [4 1]';
bas.xx     = [1 3 0]';
```

```

prob.sol.bas = bas;

% Specify the problem data.
prob.c       = [ 1 2 0]';
subi         = [1 2 2 1];
subj         = [1 1 2 3];
valij        = [1.0 1.0 1.0 1.0];
prob.a       = sparse(subi,subj,valij);
prob.blc     = [4.0 1.0]';
prob.buc     = [6.0 inf]';
prob.blx     = sparse(3,1);
prob.bux     = [];

% Use the primal simplex optimizer.
param.MSK_IPAR_OPTIMIZER = 'MSK_OPTIMIZER_PRIMAL_SIMPLEX';
[r,res] = mosekopt('minimize',prob,param)

```

Some comments:

- In the example the dual solution is defined. This is acceptable because the primal simplex optimizer is used for the reoptimization and it does not exploit a dual solution. In the future MOSEK will also contain a dual simplex optimizer and if that optimizer is used, it will be important that a “good” dual solution is specified.
- The status keys `bas.skc` and `bas.sbx` must contain only the entries BS, EQ, LL, UL, and SB. Moreover, e.g. EQ must be specified only for a fixed constraint or variable. LL and UL can be used only for a variable that has a finite lower or upper bound respectively.
- The number of constraints and variables defined to be basic must correspond exactly to the number of constraints, i.e. the row dimension of A .

7.19.2 Adding a new variable

Next, assume that the problem

$$\begin{aligned}
 & \text{minimize} && x_1 + 2x_2 - x_4 \\
 & \text{subject to} && 4 \leq x_1 + x_3 + x_4 \leq 6, \\
 & && 1 \leq x_1 + x_2, \\
 & && 0 \leq x_1, x_2, x_3, x_4.
 \end{aligned} \tag{7.45}$$

should be solved. It is identical to the problem (7.44) except that a new variable x_4 has been added. In continuation of the previous example this problem can be solved as follows (using hot-start):

```

% advs2.m. Continuation of advs1.m.

prob.c      = [prob.c; -1.0];

```

```

prob.a      = [prob.a,sparse([1.0 0.0]')];
prob.blx    = sparse(4,1);

% Reuse the old optimal basic solution.
bas         = res.sol.bas;

% Add to the status key.
bas.skx     = [res.sol.bas.skx;'LL'];

% The new variable is at it lower bound.
bas.xx      = [res.sol.bas.xx;0.0];
bas.slx     = [res.sol.bas.slx;0.0];
bas.sux     = [res.sol.bas.sux;0.0];

prob.sol.bas = bas;

[rcode,res] = mosekopt('minimize',prob,param);

% The new primal optimal solution
res.sol.bas.xx'

```

7.19.3 Fixing a variable

In e.g. branch-and-bound methods for integer programming problems it is necessary to reoptimize the problem after a variable has been fixed to a value. This can easily be achieved as follows:

```

% advs3.m. Continuation of advs2.m.

prob.blx(4) = 1;
prob.bux    = [inf inf inf 1]';

% Reuse the basis.
prob.sol.bas = res.sol.bas;

[rcode,res] = mosekopt('minimize',prob,param);

% Display the optimal solution.
res.sol.bas.xx'

```

The x_4 variable is simply fixed at the value 1 and the problem is reoptimized. Please note that the basis from the previous optimization can immediately be reused.

7.19.4 Adding a new constraint

Now, assume that the constraint

$$x_1 + x_2 \geq 2 \quad (7.46)$$

should be added to the problem and the problem should be reoptimized. The following example demonstrates how to do this.

```
% advs4.m. A continuation of advs3.m.

% Modify the problem.
prob.a      = [prob.a;sparse([1.0 1.0 0.0 0.0])];
prob.blc    = [prob.blc;2.0];
prob.buc    = [prob.buc;inf];

% Obtain the previous optimal basis.
bas         = res.sol.bas;

% Set the solution to the modified problem.
bas.skc     = [bas.skc;'BS'];
bas.xc      = [bas.xc;bas.xx(1)+bas.xx(2)];
bas.y       = [bas.y;0.0];
bas.slc     = [bas.slc;0.0];
bas.suc     = [bas.suc;0.0];

% Reuse the basis.
prob.sol.bas = bas;

% Reoptimize.
[rcode,res] = mosekopt('minimize',prob,param);

res.sol.bas.xx'
```

Please note that the slack variable corresponding to the new constraint are declared basic. This implies that the new basis is nonsingular and can be reused.

7.19.5 Using numeric values to represent status key codes

In the previous examples the constraint and variable status keys are represented using string codes. Although the status keys are easy to read they are sometimes difficult to work with in a program. Therefore, the status keys can also be represented using numeric values as demonstrated in the example:

```
% sk1.m

% Obtain all symbolic constants
% defined in MOSEK.

clear prob bas;

[r,res] = mosekopt('symbcon');
sc      = res.symbcon;

% Specify an initial basic solution.
```

```
% Please note that symbolic constants are used.
% I.e. sc.MSK_SK_LOW instead of 4.
bas.skc      = [sc.MSK_SK_LOW;sc.MSK_SK_LOW];
bas.skx      = [sc.MSK_SK_BAS;sc.MSK_SK_LOW;sc.MSK_SK_BAS];
bas.xc       = [4 1]';
bas.xx       = [1 3 0]';
prob.sol.bas = bas;

% Specify the problem data.
prob.c       = [ 1 2 0]';
subi         = [1 2 2 1];
subj         = [1 1 2 3];
valij        = [1.0 1.0 1.0 1.0];
prob.a       = sparse(subi,subj,valij);
prob.blc     = [4.0 1.0]';
prob.buc     = [6.0 inf]';
prob.blx     = sparse(3,1);
prob.bux     = [];

% Use the primal simplex optimizer.
clear param;
param.MSK_IPAR_OPTIMIZER = sc.MSK_OPTIMIZER_PRIMAL_SIMPLEX;

[r,res] = mosekopt('minimize statuskeys(1)',prob,param)

% Status keys will be numeric now i.e.

res.sol.bas.skc'

% is a vector of numeric values.
```

Please note that using the commands

```
[r,res] = mosekopt('symbcon');
sc       = res.symbcon;
```

all the symbolic constants defined within MOSEK are obtained and used in the lines

```
bas.skc = [sc.MSK_SK_LOW;sc.MSK_SK_LOW];
bas.skx = [sc.MSK_SK_BAS;sc.MSK_SK_LOW;sc.MSK_SK_BAS];
```

These two lines are in fact equivalent to

```
bas.skc = [1;1];
bas.skx = [3;1;3];
```

However, it is **not** recommended to specify the constraint and variable status keys this way because it is less readable and portable. Indeed if e.g. MOSEK later changes the definition that 1 is equivalent to 'LL', all programs using numerical keys will be incorrect whereas using the symbolic constants the programs remain correct.

7.20 Using names

In MOSEK it is possible to give the objective, each constraint, each variable, and each cone a name. In general such names are not really needed except in connection with reading and writing MPS files. See Section 7.21 for details.

All the names are specified in the `prob.names` structure.

```
% The problem is named.
prob.names.name = 'CQ0 example';

% Objective name.
prob.names.obj = 'cost';

% The two constraints are named.
prob.names.con{1} = 'constraint_1';
prob.names.con{2} = 'constraint_2';

% The six variables are named.
prob.names.var = cell(6,1);
for j=1:6
    prob.names.var{j} = sprintf('x%d',j);
end

% Finally the two cones are named.
prob.names.cone{1} = 'cone_a';
prob.names.cone{2} = 'cone_b';
```

7.20.1 Blanks in names

Although it is allowed to use blanks (spaces) in names it is not recommended to do so except for the problem name. In general, avoid names like “x 1” or “con 1”.

7.21 MPS files

An industry standard format for storing linear optimization problems in an ASCII file is the so-called MPS format. For readers not familiar with the MPS format a specification of the MPS format supported by MOSEK can be seen in Appendix A.

The advantage of the MPS format is that problems stored in this format can be read by any commercial optimization software, so it facilitates communication of optimization problems.

7.21.1 Reading an MPS file

It is possible to use `mosekopt` to read an MPS file containing the problem data. In this case `mosekopt` reads data from an MPS file and returns both the problem data and the optimal

solution, if required. Assume that `afiro.mps` is the MPS file from which `mosekopt` should read the problem data, then this task is performed using the command

```
[r,res] = mosekopt('read(afiro.mps)');
```

In this case `res.prob` will contain several fields with the problem data. E.g.

```
res.prob.c'
```

will display the c -vector.

The names used in the MPS file is also available in the `prob.names` structure.

```
% All names.
```

```
prob.names
```

```
% Constraint names.
```

```
prob.names.con
```

The quadratic terms of a problem can be accessed and displayed in a similar manner:

```
% mpsrd.m

% Read data from the file wp12-20.mps.

[r,res] = mosekopt('read(wp12-20.mps)');

% Looking at the problem data
prob = res.prob;
clear res;

% Form the quadratic term in the objective.
q = sparse(prob.qosubi,prob.qosubj,prob.qoval);

% Get a graphical picture.
spy(q) % Notice that only the lower triangular part is defined.
```

7.21.2 Writing a MPS files

It is possible to write an MPS file using MOSEK. To write a problem contained in a MATLAB structure `prob` to the file “`datafile.mps`”, use the command:

```
% Write the data defined by prob to an MPS file
```

```
% named datafile.mps
```

```
mosekopt('write(datafile.mps)',prob);
```

If the `prob.names` field is defined, MOSEK will use those names when writing the MPS file, otherwise MOSEK will use generic (automatically generated) names.

7.22 User call-back functions

A call-back function is a user-defined MATLAB function to be called by MOSEK on a given event. The optimization toolbox supports two types of call-back functions which are presented below.

7.22.1 Log printing via call-back function

When using `mosekopt` it is possible to control the amount of information that `mosekopt` prints to the screen, e.g.

```
[r,res] = mosekopt('minimize echo(0)',prob)
```

forces `mosekopt` to not print log information — the string `echo(0)` indicates that no output should be printed during optimization. A high value in the `echo(n)` command, e.g. `echo(3)`, forces MOSEK to display more log information.

It is possible to redirect the MOSEK log printing almost anywhere using a user-defined log call-back function. It works as follows. Create an m-file to handle the log output, similar to:

```
function myprint(handle,str)
% handle: Is user defined data structure
% str    : Is a log string.
%
fprintf(handle,'%s',str);
```

The name and actions of the function are not important, but its argument list must be identical to the example: It must accept two arguments. The first argument, `handle`, is a user-defined MATLAB structure and the second argument, `str`, is a text string. In the example above `myprint` prints the string to a file defined by `handle`.

The following code fragment shows how to tell MOSEK to send log output to the `myprint` function.

```
%
% In this example the MOSEK log info
% should be printed to the screen and to a file named
% mosek.log.
%
fid          = fopen('mosek.log','wt');
callback.log = 'myprint';
callback.loghandle = fid;

%
% The 'handle' argument in myprint() will be identical to
% callback.loghandle when called.
```

```
%
mosekopt('minimize',prob,[],callback);
```

7.22.2 The iteration call-back function

It is possible to specify a function to be called frequently during the optimization. Typically this call-back function is used to display information about the optimization process or to terminate it.

The iteration call-back function has the following form:

```
function [r] = myiter(handle,where,info)
% handle: Is a user-defined data structure.
% where : Is an integer indicating from where in the optimization
%         process the callback was invoked.
% info  : A MATLAB structure containing information about the state of the
%         optimization.

r = 0; % r should always be assigned a value.

if handle.symbcon.MSK_CALLBACK_BEGIN_INTPNT==where
    fprintf('Interior point optimizer started\n');
end

% Print primal objective
fprintf('Interior-point primal obj.: %e\n',info.MSK_DINF_INTPNT_PRIMAL_OBJ);

% Terminate when cputime > handle.maxtime
if info.MSK_DINF_INTPNT_TIME > handle.maxtime
    r = 1;
else
    r = 0;
end

if handle.symbcon.MSK_CALLBACK_END_INTPNT==where
    fprintf('Interior-point optimizer terminated\n');
end
```

The function accepts three arguments: The first argument, **handle**, is a user-defined MATLAB structure, the second argument, **where**, indicates from where in the optimization process the call-back was invoked and the third argument, **info**, is a structure containing information about the process. For details about **info** see Section 8.1.7g If the function returns a non-zero value, MOSEK will terminate the optimization process immediately.

In order to inform MOSEK about the iteration call-back function the fields **iter** and **iterhandle** are initialized as shown in the following example.

```
[r,res] = mosekopt('symbcon');
data.maxtime = 100.0;
data.symbcon = res.symbcon;

callback.iter = 'myiter';
callback.iterhandle = data;

mosekopt('minimize',prob,[],callback);
```

7.23 The license system

By default a license token remains checked out for the duration of the matlab session. This can be changed such that the license is returned after each call to mosek by setting the parameter **MSK_IPAR_CACHE_LICENSE**.

```
param.MSK_IPAR_CACHE_LICENSE = 'MSK_OFF'; %set parameter.
[r,res] = mosekopt('minimize',prob,param); %call mosek.
```

By default an error will be returned if no license token is available. By setting the parameter **MSK_IPAR_LICENSE_WAIT** mosek can be instructed to wait until a license token is available.

```
param.MSK_IPAR_LICENSE_WAIT = 'MSK_ON'; %set parameter.
[r,res] = mosekopt('minimize',prob,param); %call mosek.
```


Chapter 8

Command reference

After studying the examples presented in the previous chapter, it should be possible to use most of the facilities in the MOSEK optimization toolbox. A more formal specification of the main data structures employed by MOSEK and a command reference are provided in this chapter.

8.1 Data structures

In each of the subsequent sections the most important data structures employed by MOSEK are discussed.

8.1.1 prob

Description: The `prob` data structure is used to communicate an optimization problem to MOSEK or for MOSEK to return an optimization problem to the user. It defines an optimization problem using a number of subfields.

Subfields:

<code>.names</code>	A MATLAB structure which contains the problem name, the name of the objective, and so forth. See Section 8.1.2 .
<code>.qosubi</code>	i subscript for element q_{ij}^o in Q^o . See (8.6) .
<code>.qosubj</code>	j subscript for element q_{ij}^o in Q^o . See (8.6) .
<code>.qoval</code>	Numerical value for element q_{ij}^o in Q^o . See (8.6) .
<code>.qcsubk</code>	k subscript for element q_{ij}^p in Q^p . See (8.7) .
<code>.qcsubi</code>	i subscript for element q_{ij}^p in Q^p . See (8.7) .
<code>.qcsubj</code>	j subscript for element q_{ij}^p in Q^p . See (8.7) .
<code>.qcval</code>	Numerical value for element q_{ij}^p in Q^p . See (8.7) .

<code>.c</code>	Linear term in the objective.
<code>.a</code>	The constraint matrix. It must be a sparse matrix having the number of rows and columns equivalent to the number of constraints and variables in the problem. This field should always be defined, even if the problem does not have any constraints. In that case a sparse matrix having zero rows and the correct number of columns is the appropriate definition of the field.
<code>.blc</code>	Lower bounds of the constraints. $-\infty$ denotes an infinite lower bound. If the field is not defined or <code>blc==[]</code> , then all the lower bounds are assumed to be equal to $-\infty$.
<code>.buc</code>	Upper bounds of the constraints. ∞ denotes an infinite upper bound. If the field is not defined or <code>buc==[]</code> , then all the upper bounds are assumed to be equal to ∞ .
<code>.blx</code>	Lower bounds on the variables. $-\infty$ denotes an infinite lower bound. If the field is not defined or <code>blx==[]</code> , then all the lower bounds are assumed to be equal to $-\infty$.
<code>.bux</code>	Upper bounds on the variables. ∞ denotes an infinite upper bound. If the field is not defined or <code>bux==[]</code> , then all the upper bounds are assumed to be equal to ∞ .
<code>.ints</code>	A MATLAB structure which has the subfields <code>.sub; % Required.</code> <code>.pri; % Subfields.</code> <code>ints.sub</code> is a one-dimensional array containing the indexes of the integer-constrained variables. Hence, <code>ints.sub</code> is identical to the set \mathcal{J} in (8.5). <code>ints.pri</code> is also a one dimensional array of the same length as <code>ints.sub</code> . The <code>ints.pri(k)</code> is the branching priority assigned to variable index <code>ints.sub(k)</code> .
<code>.cones</code>	A MATLAB cell array defining the conic constraints (8.4). See Section 8.1.3 for details on this structure.
<code>.sol</code>	A MATLAB structure containing a guess on the optimal solution which some of the optimizers in MOSEK may exploit. See Section 8.1.4 for details on this structure.
<code>.prisen</code>	A MATLAB structure which has the subfields: <code>.cons.subu</code> Indexes of constraints, where upper bounds are analyzed for sensitivity. <code>.cons.subl</code> Indexes of constraints, where lower bounds are analyzed for sensitivity.

<code>.vars.subu</code>	Indexes of variables, where upper bounds are analyzed for sensitivity.
<code>.vars.subl</code>	Indexes of variables, where lower bounds are analyzed for sensitivity.
<code>.sub</code>	Index of variables where coefficients are analysed for sensitivity.

Comments:

MOSEK solves an optimization problem which has the form of minimizing or maximizing an objective function

$$\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n q_{ij}^o x_i x_j + c_j x_j \quad (8.1)$$

subject to the functional constraints

$$l_k^c \leq \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n q_{ij}^p x_i x_j + \sum_{j=1}^n a_{kj} x_j \leq u_k^c, \quad k = 1, \dots, m, \quad (8.2)$$

the variable bound constraints

$$l_j^x \leq x_j \leq u_j^x, \quad j = 1, \dots, n \quad (8.3)$$

and the conic constraint

$$x \in \mathcal{C}. \quad (8.4)$$

Finally, some variables may be integer-constrained, i.e.

$$x_j \text{ integer-constrained for all } j \in \mathcal{J} \quad (8.5)$$

where x is the decision variables and all the other quantities are the parameters of the problem and they are presented below:

Q^o : The quadratic terms $q_{ij}^o x_i x_j$ in the objective are stored in the matrix Q^o as follows

$$Q^o = \begin{bmatrix} q_{11}^o & \cdots & q_{1n}^o \\ \vdots & \cdots & \vdots \\ q_{n1}^o & \cdots & q_{nn}^o \end{bmatrix}.$$

In MOSEK it is assumed that Q^o is symmetric, i.e.

$$q_{ij}^o = q_{ji}^o$$

and therefore only the lower triangular part in Q^o should be specified.

c : It is the linear part of the objective specifying the c_j in the linear term $c_j x_j$.

Q^p : The quadratic terms $q_{ij}^p x_i x_j$ in the k th constraint are stored in the Q^p matrix as follows

$$Q^p = \begin{bmatrix} q_{11}^p & \cdots & q_{1n}^p \\ \vdots & \cdots & \vdots \\ q_{n1}^p & \cdots & q_{nn}^p \end{bmatrix}.$$

MOSEK assumes that Q^p is symmetric, i.e.

$$q_{ij}^p = q_{ji}^p$$

and therefore only the lower triangular part in Q^p should be specified.

A : The constraint matrix A is given by

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \cdots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}.$$

In MOSEK it is assumed that A is a sparse matrix, i.e. most of the coefficients in A are zero. Therefore, only non-zeros elements in A are stored and worked with. This usually saves a lot of storage and speeds up the computations.

l^c : Specifies the lower bounds of the constraints.

u^c : Specifies the upper bounds of the constraints.

l^x : Specifies the lower bounds on the variables.

u^x : Specifies the upper bounds on the variables.

$cones$: Specifies the conic constraint. Let

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

be vectors comprised of disjointed subsets of the decision variables x (each decision variable is a member of exactly one x^t), e.g.

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

Next, define

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

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:

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

– Rotated quadratic cone:

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

All the parameters of the optimization problem are stored using one or more subfields of the **prob** structure using the naming convention in Table 8.1.

Field name	Type	Dimension	Optional	Problem parameter
qosubi	int	length(qoval)	Yes	q_{ij}^o
qosubj	int	length(qoval)	Yes	q_{ij}^o
qoval	double	length(qoval)	Yes	q_{ij}^o
c	double	n	Yes	c_j
qcsubk	int	length(qcval)	Yes	q_{ij}^p
qcsubi	int	length(qcval)	Yes	q_{ij}^p
qcsubj	int	length(qcval)	Yes	q_{ij}^p
qcval	double	length(qcval)	Yes	q_{ij}^p
a	Sparse matrix	$m \times n$	No	a_{ij}
blc	double	m	Yes	l_k^c
buc	double	m	Yes	u_k^c
blx	double	n	Yes	l_k^x
bux	double	n	Yes	u_k^x
ints	MATLAB structure	$ \mathcal{J} $	Yes	\mathcal{J}
cones	MATLAB cell array	k	Yes	\mathcal{C}

Table 8.1: The relation between fields and problem parameters.

In Table 8.1 all the parameters are listed with their corresponding type. The **int** type indicates that the field must contain an integer value, **double** indicates a real number. The relationship between Q^o and Q^p and the subfields of the **prob** structure is as follows:

- The quadratic terms in the objective:

$$q_{\text{qosubi}(\mathbf{t}), \text{qoval}(\mathbf{t})}^o = \text{qoval}(\mathbf{t}), \ t = 1, 2, \dots, \text{length}(\text{qoval}). \quad (8.6)$$

Since Q^o by assumption is symmetric, all elements are assumed to belong to the lower triangular part. If an element is specified multiple times, the different elements are added together.

- The quadratic terms in the constraints:

$$q_{qcsubi(t),qcsbj(t)}^{qcsbk(t)} = qcval(t), \quad t = 1, 2, \dots, \text{length}(qcval). \quad (8.7)$$

Since Q^p by assumption is symmetric, all elements are assumed to belong to the lower triangular part. If an element is specified multiple times, the different elements are added together.

8.1.2 names

This structure is used to store all the names of individual items in the optimization problem such as the constraints and the variables. The structure contains the subfields:

<code>.name</code>	Contains the problem name.
<code>.obj</code>	Contains the name of the objective.
<code>.con</code>	Is a MATLAB cell array where <code>names.con{i}</code> contains the name of the i th constraint.
<code>.var</code>	Is a MATLAB cell array where <code>names.var{j}</code> contains the name of the j th constraint.
<code>.cone</code>	Is a MATLAB cell array where <code>names.cone{t}</code> contains the name of the t th conic constraint.

8.1.3 cones

`cones` is a MATLAB cell array containing one structure per cone in the optimization problem, i.e. `cones{t}` is used to specify the t th cone in the optimization problem.

The structure contains the subfields:

<code>.type</code>	<code>cones{t}.type</code> contains the cone type for the t th cone. The <code>type</code> subfield has one of the values 'MSK_CT_QUAD' or 'MSK_CT_RQUAD', indicating if cone is a quadratic cone or a rotated quadratic cone.
<code>.sub</code>	<code>cones{t}.sub</code> is a list of variable indexes specifying which variables are members of the cone.

8.1.4 sol

Description: A MATLAB structure used to store one or more solutions to an optimization problem. The structure has one subfield for each possible solution type.

Subfields:

<code>.itr</code>	Interior (point) solution computed by the interior-point optimizer.
<code>.bas</code>	Basic solution computed by the simplex optimizers and basis identification procedure.
<code>.int</code>	Integer solution computed by the mixed-integer optimizer.

Comments: Each of the solutions `sol.itr`, `sol.bas`, and `sol.int` may contain one or more of the fields:

<code>.prosta</code>	Problem status. See Appendix F.38.
<code>.solsta</code>	Solution status. See Appendix F.51.
<code>.skc</code>	Constraint status keys. See Table 7.1.
<code>.skx</code>	Variable status keys. See Table 7.1.
<code>.skn</code>	Conic status keys. See Section 7.1.
<code>.xc</code>	Constraint activities, i.e. $x_c = Ax$ where x is the optimal solution.
<code>.xx</code>	Variable activities, i.e. the optimal x solution.
<code>.y</code>	Identical to <code>sol.slc-sol.suc</code> .
<code>.slc</code>	Dual solution corresponding to the lower constraint bounds.
<code>.suc</code>	Dual solution corresponding to the upper constraint bounds.
<code>.slx</code>	Dual solution corresponding to the lower variable bounds.
<code>.sux</code>	Dual solution corresponding to the upper variable bounds.
<code>.snx</code>	Dual solution corresponding to the conic constraint.
<code>.pobjval</code>	The primal objective value.

The fields `.skn` and `.snx` cannot occur in the `.bas` and `.int` solutions. In addition the fields `.y`, `.slc`, `.suc`, `.slx`, and `.sux` cannot occur in the `.int` solution since integer problems does not have a well-defined dual problem, and hence no dual solution.

8.1.5 prisen

Description:

Results of the primal sensitivity analysis.

Subfields:

.cons	MATLAB structure with the subfields:
.lr_bl	Left value β_1 in the linearity interval for a lower bound.
.rr_bl	Right value β_2 in the linearity interval for a lower bound.
.ls_bl	Left shadow price s_l for a lower bound.
.rs_bl	Right shadow price s_r for a lower bound.
.lr_bu	Left value β_1 in the linearity interval for an upper bound.
.rr_bu	Right value β_2 in the linearity interval for an upper bound.
.ls_bu	Left shadow price s_l for an upper bound.
.rs_bu	Right shadow price s_r for an upper bound.
.var	MATLAB structure with the subfields:
.lr_bl	Left value β_1 in the linearity interval for a lower bound on a variable.
.rr_bl	Right value β_2 in the linearity interval for a lower bound on a variable.
.ls_bl	Left shadow price s_l for a lower bound on a variable.
.rs_bl	Right shadow price s_r for a lower bound on a variable.
.lr_bu	Left value β_1 in the linearity interval for an upper bound on a variable.
.rr_bu	Right value β_2 in the linearity interval for an upper bound on a variable.
.ls_bu	Left shadow price s_l for an upper bound on a variable.
.rs_bu	Right shadow price s_r for an upper bound on a variable.

8.1.6 duasen

Description:

Results of dual the sensitivity analysis.

Subfields:

.lr_c	Left value β_1 in linearity interval for an objective coefficient.
.rr_c	Right value β_2 in linearity interval for an objective coefficient.
.ls_c	Left shadow price s_l for an objective coefficient.
.rs_c	Right shadow price s_r for an objective coefficient.

8.1.7 info

`info` is a MATLAB structure containing a subfield for each item in the MOSEK optimization task database, e.g. the `info.MSK_DINF_BI_CPUTIME` field specifies the amount of time spent in the basis identification in the last optimization. In Sections F.13 and F.17 all the items in the task information database are listed.

8.1.8 symbcon

`symbcon` is a MATLAB structure containing a subfield for each MOSEK symbolic constant, e.g. the field `symbcon.MSK_DINF_BI_CPUTIME` specifies the value of the symbolic constant “MSK_DINF_BI_CPUTIME”. In Appendix F all the symbolic constants are listed.

8.1.9 callback

`callback` A MATLAB structure containing the subfields (all of them are optional):

<code>.loghandle</code>	A MATLAB data structure or just <code>[]</code> .
<code>.log</code>	The name of a user-defined function which must accept two input arguments, e.g. <pre>function myfunc(handle,str)</pre> <p>where <code>handle</code> will be identical to <code>callback.handle</code> when <code>myfunc</code> is called, and <code>str</code> is a string of text from the log file.</p>
<code>.iterhandle</code>	A MATLAB data structure or just <code>[]</code> .
<code>.iter</code>	The name of a user-defined function which must accept three input arguments, e.g. <pre>function myfunc(handle,where,info)</pre> <p>where <code>handle</code> will be identical to <code>callback.iterhandle</code> when <code>myfunc</code> is called, <code>where</code> indicates the current progress of the colver and <code>info</code> is the current information database. See 8.1.7 for further details about the <code>info</code> data structure.</p>

8.2 An example of a command reference

All functions are documented using the format:

- `somefunction`

Description: The purpose of the function.

Syntax:

```
[ret1,ret2] = somefunction(arg1,arg2)
```

Arguments:

<code>arg1</code>	A description of this argument.
<code>arg2</code>	(Optional) A description of this argument which is optional. However, if argument 3 is specified, this argument must be specified too.
<code>arg3</code>	Another useful argument.

Returns:

<code>ret1</code>	A description of the first return <code>ret1</code> .
<code>ret2</code>	(Optional) A description of the second return <code>ret2</code> .

Comments:

Potentially some comments about the function.

Examples:

Some examples of the use of the function.

8.3 Functions provided by the MOSEK optimization toolbox

- `mosekopt`

Description Solves an optimization problem. Data specifying the optimization problem can either be read from a file or be inputted directly from MATLAB. It is also possible to write a file using `mosekopt`.

Syntax:

```
[rcode,res] = mosekopt(cmd,prob,param,callback)
```

Arguments:

<code>cmd</code>	<code>cmd</code> is a string containing commands to MOSEK on what to do. E.g. the string 'minimize info' means that the objective should be minimized, and information about the optimization process should be returned in <code>res.info</code> . The following commands are recognized by <code>mosekopt</code> :
------------------	--

<code>aformat(n)</code>	<p>If the problem data are read from a file, this command controls the format <code>res.prob.a</code> upon return. For <code>aformat(0)</code> <code>res.prob.a</code> is a sparse matrix, and for <code>aformat(1)</code> the constraint matrix is given in coordinate format in which case the sparse matrix can be constructed as</p> <pre> m = % number of constraints n = % number of variables a = sparse(res.prob.a.subi,... res.prob.a.subj,... res.prob.a.val,... m,n); </pre>
<code>anapro</code>	Runs the problem analyzer.
<code>echo(n)</code>	Controls how much information is echoed to the screen. Here, <code>n</code> must be a nonnegative integer, where 0 means that no information is displayed and 3 means that all information is displayed.
<code>info</code>	Return the complete task information database in <code>res.info</code> . This database contains various task specific information. See Section 8.1.7 for details the <code>info</code> data structure.
<code>param</code>	Return the complete parameter database in <code>res.param</code> .
<code>maximize</code>	Maximize the objective.
<code>minimize</code>	Minimize the objective.
<code>nokeepenv</code>	Delete the MOSEK environment after each run. This can increase the license checkout overhead significantly and is therefore only intended as a debug feature.
<code>read(name)</code>	Request that data is read from a file “ <code>name</code> ”.
<code>statuskeys(n)</code>	Controls the format of status keys (problem status, solution status etc.) in the returned problem, where <code>statuskeys(0)</code> means that all the status keys are returned as strings, and <code>statuskeys(1)</code> means that all the status keys are returned as numeric codes.
<code>symbcon</code>	Return the <code>symbcon</code> data structure in <code>res.symbcon</code> . See Section 8.1.8 for details on the <code>symbcon</code> data structure.
<code>write(name),write()</code>	Write problem to the file “ <code>name</code> ”.

prob	(Optional) A MATLAB structure containing the problem data. See Table 8.1 for details.
param	(Optional) A MATLAB structure which is used to specify algorithmic parameters to MOSEK. The fields of param must be valid MOSEK parameter names. Moreover, the values corresponding to the fields must be of a valid type, i.e. the value of a string parameter must be a string, the value of an integer parameter must be an integer etc.
callback	(Optional) A MATLAB structure defining call-back data and functions. See Sections 7.22 and 8.1.9 for details.

Returns:

rcode	Return code. The interpretation of the value of the return code is listed in Appendix F.
res	(Optional) Solution obtained by the interior-point algorithm.
.sol	The data structure of the type sol is discussed in Section 8.1.
.info	A MATLAB structure containing the task information database which contains various task related information such as the number of iterations used to solve the problem. However, this field is only defined if info appeared in the cmd command when mosekopt is invoked.
.param	A MATLAB structure which contain the complete MOSEK parameter database. However, this field is defined only if the param command is present in cmd when mosekopt is invoked.
.prob	Contains the problem data if the problem data was read from a file.

Examples:

The following example

demonstrates how to display the MOSEK parameter database and how to use the primal simplex optimizer instead of the default optimizer.

```
% Obtain the parameter database.
[rcode,res] = mosekopt('param');

% Display the parameter database.
res.param
param = res.param;

% Modify a parameter.
param.MSK_IPAR_OPTIMIZER = 3
```

```
% Optimize a problem.
[rcode,res]=mosekopt('minimize info',prob,param);

% Display the information database.
res.info
```

- msklpopt
- mskqpopt
- mskenopt
- mskgpopt
- mskscopt

Description These functions provide an easy-to-use but less flexible interface than the `mosekopt` function. In fact these procedures are just wrappers around the `mosekopt` interface and they are defined in MATLAB m-files.

Syntax:

```
res = msklpopt(c,a,blc,buc,blx,bux,param,cmd);
res = mskqpopt(q,c,a,blc,buc,blx,bux,param,cmd);
res = mskenopt(d,c,a,blc,buc,blx,bux,param,cmd);
res = mskgpopt(d,a,map,param,cmd);
res = mskscopt(opr,opri,oprj,oprj,oprj,oprj,...
               c,a,blc,buc,blx,bux,param,cmd)
```

Arguments: For a description of the arguments we refer the reader to the actual m-files stored in

```
<root>\mosek\6\toolbox\matlab\solvers
```

Please note that the MATLAB command `help`, e.g.

```
help msklpopt
```

will produce some usage information for the functions.

Returns:

Identical to the `res` structure returned by `mosekopt`.

- mskgpwri

Description This function writes a Geometric Programming (gp) problem to a file in a format compatible with the `mskexpopt` command line tool.

Syntax:

```
res = mskgpwri(c,a,map,filename)
```

Arguments:

`c,a,map` Data in the same format accepted by `mskgpopt`.
`filename` The output file name.

Returns:

Nothing.

- `mskgpread`

Description This function reads a Geometric Programming (gp) problem from a file compatible with the `mskexpopt` command line tool.

Syntax:

```
[c,a,map] = mskgpread (filename)
```

Arguments:

`filename` The name of the file to read.

Returns:

`c,a,map` Data in the same format accepted by `mskgpopt`.

8.4 MATLAB optimization toolbox compatible functions

The functions presented in this section intends to provide compatibility with the corresponding MATLAB optimization toolbox functions. Although they are not perfectly compatible, the differences usually do not cause any problems.

8.4.1 Linear and quadratic optimization

- `linprog`

Description

Solves the linear optimization problem:

$$\begin{aligned} &\text{minimize} && f^T x \\ &\text{subject to} && Ax \leq b, \\ & && Bx = c, \\ & && l \leq x \leq u. \end{aligned}$$

Syntax:

```
[x,fval,exitflag,output,lambda]
= linprog(f,A,b,B,c,l,u,x0,options)
```

Arguments:

f	The objective function.
A	Constraint matrix for the less-than equal inequalities. Use $A = []$ if there are no inequalities.
b	Right-hand side for the less-than equal inequalities. Use $b = []$ if there are no inequalities.
B	(Optional) Constraint matrix for the equalities. Use $B = []$ if there are no equalities.
c	(Optional) Right-hand side for the equalities. Use $c = []$ if there are no equalities.
l	(Optional) Lower bounds on the variables. Please use $-\infty$ to represent infinite lower bounds.
u	(Optional) Upper bounds on the variables. Please use ∞ to represent infinite upper bounds.
x0	(Optional) An initial guess for the starting point. This information is ignored by MOSEK.
options	(Optional) An optimization options structure. See the <code>optimset</code> function for the definition of the optimization options structure. <code>linprog</code> uses the options <code>.Diagnostics</code> <code>.Display</code> <code>.MaxIter</code>

Returns:

x	The optimal x solution.
fval	The optimal objective value, i.e. $f^T x$.
exitflag	A number which has the interpretation: < 0 The problem is likely to be either primal or dual infeasible. $= 0$ The maximum number of iterations was reached. > 0 x is an optimal solution.
output	<code>.iterations</code> Number of iterations spent to reach the optimum. <code>.algorithm</code> Always defined as 'large-scale: interior-point'.
lambda	<code>.lower</code> Lagrange multipliers for lower bounds l . <code>.upper</code> Lagrange multipliers for upper bounds u . <code>.ineqlin</code> Lagrange multipliers for the inequalities. <code>.eqlin</code> Lagrange multipliers for the equalities.

Examples:

```
% Optimize a problem having only linear inequalities.
x = linprog(f,A,b);
```

- **quadprog**

Description

Solves the quadratic optimization problem:

$$\begin{aligned}
 &\text{minimize} && \frac{1}{2}x^T Hx + f^T x \\
 &\text{subject to} && Ax \leq b, \\
 & && Bx = c, \\
 & && l \leq x \leq u.
 \end{aligned} \tag{8.8}$$

Syntax:

```
[x,fval,exitflag,output,lambda]
= quadprog(H,f,A,b,B,c,l,u,x0,options)
```

Arguments:

H	Hessian of the objective function. H must be a symmetric matrix. Contrary to the MATLAB optimization toolbox, MOSEK handles only the cases where H is positive semi-definite. On the other hand MOSEK always computes a global optimum, i.e. the objective function has to be strictly convex.
f	See (8.8) for the definition.
A	Constraint matrix for the less-than equal inequalities. Use $A = []$ if there are no inequalities.
b	Right-hand side for the less-than equal inequalities. Use $b = []$ if there are no inequalities.
B	(Optional) Constraint matrix for the equalities. Use $B = []$ if there are no equalities.
c	(Optional) Right-hand side for the equalities. Use $c = []$ if there are no equalities.
l	(Optional) Lower bounds on the variables. Please use $-\infty$ to represent infinite lower bounds.
u	(Optional) Upper bounds on the variables. Please use ∞ to represent infinite upper bounds.
x0	(Optional) An initial guess for the starting point. This information is ignored by MOSEK.

options (Optional) An optimization options structure. See the `optimset` function for the definition of the optimizations options structure. `quadprog` uses the options

- .Diagnostics
- .Display
- .MaxIter
- .Write

Returns:

`x` The x solution.

`fval` The optimal objective value i.e. $\frac{1}{2}x^T Hx + f^T x$.

`exitflag` A scalar which has the interpretation:

- < 0 The problem is likely to be either primal or dual infeasible.
- $= 0$ The maximum number of iterations was reached.
- > 0 x is an optimal solution.

`output` .iterations Number of iterations spent to reach the optimum.

.algorithm Always defined as 'large-scale: interior-point'.

`lambda` .lower Lagrange multipliers for lower bounds l .

.upper Lagrange multipliers for upper bounds u .

.ineqlin Lagrange multipliers for inequalities.

.eqlin Lagrange multipliers for equalities.

Examples:

```
% Optimizes problem only
% having linear inequalities.
x = quadprog(H,f,A,b);
```

8.4.2 For linear least squares problems

- lsqlin

Description

Solves the linear least squares problem:

$$\begin{aligned}
 &\text{minimize} && \frac{1}{2} \|Cx - d\|_2^2 \\
 &\text{subject to} && Ax \leq b, \\
 &&& Bx = c, \\
 &&& l \leq x \leq u.
 \end{aligned} \tag{8.9}$$

Syntax:

```
[x,resnorm,residual,exitflag,output,lambda]
= lsqlin(C,d,A,b,B,c,l,u,x0,options)
```

Arguments:

C	A matrix. See problem (8.9) for the purpose of the argument.
d	A vector. See problem (8.9) for the purpose of the argument.
A	Constraint matrix for the less-than equal inequalities. Use $A = []$ if there are no inequalities.
b	Right-hand side for the less-than equal inequalities. Use $b = []$ if there are no inequalities.
B	(Optional) Constraint matrix for the equalities. Use $B = []$ if there are no equalities.
c	(Optional) Right-hand side for the equalities. Use $c = []$ if there are no equalities.
l	(Optional) Lower bounds on the variables. Please use $-\infty$ to represent infinite lower bounds.
u	(Optional) Upper bounds on the variables. Please use ∞ to represent infinite lower bounds.
x0	(Optional) An initial guess for the starting point. This information is ignored by MOSEK.
options	(Optional) An optimization options structure. See the function <code>optimset</code> function for the definition of the optimization options structure. <code>lsqprog</code> uses the options <code>.Diagnostics</code> <code>.Display</code> <code>.MaxIter</code>

Returns:

x	The optimal x solution.
resnorm	The squared norm of the optimal residuals, i.e. $\ Cx - d\ ^2$ evaluated at the optimal solution.
residual	The residual $Cx - d$.
exitflag	A scalar which has the interpretation: < 0 The problem is likely to be either primal or dual infeasible. $= 0$ The maximum number of iterations was reached. > 0 x is the optimal solution.

output	.iterations	Number of iterations spent to reach the optimum.
	.algorithm	Always defined as 'large-scale: interior-point'.
lambda	.lower	Lagrange multipliers for lower bounds l .
	.upper	Lagrange multipliers for upper bounds u .
	.ineqlin	Lagrange multipliers for inequalities.
	.eqlin	Lagrange multipliers for equalities.

Examples:

```
% Solve a linear least
% squares problem.
x = lsqqlin(C,d,A,b);
```

• **lsqnonneg****Description**

Solves the linear least squares problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2} \|Cx - d\|_2^2 \\ \text{subject to} & x \geq 0. \end{array} \quad (8.10)$$

Syntax:

```
[x,resnorm,residual,exitflag,output,lambda]
= lsqnonneg(C,d,x0,options)
```

Arguments:

C	See problem (8.10).
d	See problem (8.10).
x0	(Optional) An initial guess for the starting point. This information is ignored by MOSEK.
options	(Optional) An optimizations options structure. See the <code>optimset</code> function for the definition of the optimization options structure. <code>lsqqlin</code> uses the options .Diagnostics .Display .MaxIter

Returns:

x	The x solution.
resnorm	The squared norm of the optimal residuals, i.e.

$$\|Cx - d\|^2$$

evaluated at the optimal solution.

residual		The residual $Cx - d$.
exitflag		A number which has the interpretation:
	< 0	The problem is likely to be either primal or dual infeasible.
	$= 0$	The maximum number of iterations was reached.
	> 0	x is optimal solution.
output	.iterations	Number of iterations spend to reach the optimum.
	.algorithm	Always defined to be 'large-scale: interior-point'.
lambda	.lower	Lagrange multipliers for lower bounds l .
	.upper	Lagrange multipliers for upper bounds u .
	.ineqlin	Lagrange multipliers for inequalities.
	.eqlin	Lagrange multipliers for equalities.

Comments:

This procedure just provides an easy interface to `lsqlin`. Indeed all the procedure does is to call `lsqlin` with the appropriate arguments.

Examples:

```
% Solve the problem
x = lsqnonneg(C,d);
```

8.4.3 The optimization options

The procedures in the optimization toolbox accepts some options controlling e.g. the amount of information displayed or the stopping criterion.

In general, due to the fact that MOSEK and MATLAB optimization toolboxes employ different algorithms the toolboxes use different options. Therefore, the MOSEK optimization toolbox ignores most of the options recognized by the MATLAB toolbox. The description of the `optimset` function lists which MATLAB options MOSEK recognizes.

8.4.3.1 Viewing and modifying the optimization options

- `optimget`

Description

Obtains a value of an optimization parameter.

Syntax:

```
val = optimget(options,param,default)
```

Arguments:

<code>options</code>	The optimization options structure.
<code>param</code>	Name of the optimization parameter for which the value should be obtained.
<code>default</code>	(Optional) If <code>param</code> is not defined, the value of <code>default</code> is returned instead.

Returns:

<code>val</code>	Value of the required option. If the option does not exist, then <code>[]</code> is returned unless the value <code>'default'</code> is defined in which case the default value is returned.
------------------	--

Comments: See the `optimset` function for which parameters that can be set.

Examples:

```
% Obtain the value of the diagnostics
% option.
val = optimget(options,'Diagnostics');

% val is equal to the default value.
val = optimget(options,'Nopar',1.0e-1);
```

- `optimset`

Description

Obtains and modifies the optimization options structure. Only a subset of the fields in the optimization structure recognized by the MATLAB optimization toolbox is recognized by MOSEK.

In addition the optimization options structure can be used to modify all the MOSEK specific parameters defined in Appendix E.

<code>.Diagnostics</code>	Used to control how much diagnostic information is printed. Following values are accepted:
<code>off</code>	No diagnostic information is printed.
<code>on</code>	Diagnostic information is printed.
<code>.Display</code>	Defines what information is displayed. The following values are accepted:
<code>off</code>	No output is displayed.
<code>iter</code>	Some output is displayed for each iteration.
<code>final</code>	Only the final output is displayed.
<code>.MaxIter</code>	Maximum number of iterations allowed.
<code>.Write</code>	A filename to write the problem to. If equal to the empty string no file is written. E.g the option

`Write(myfile.opf)`

writes the file `myfile.opf` in the `opf` format.

Syntax:

```
options = optimset(arg1,arg2,
                  param1,value1,
                  param2,value2,...)
```

Arguments:

<code>arg1</code>	(Optional) Is allowed to be any of the following two things: Any string The same as using no argument. A structure The argument is assumed to be a structure containing options, which are copied to the return options.
<code>param1</code>	(Optional) A string containing the name of a parameter that should be modified.
<code>value1</code>	(Optional) The new value assigned to the parameter with the name <code>param1</code> .
<code>param2</code>	(Optional) Has the same interpretation as <code>param1</code> .
<code>value2</code>	(Optional) Has the same interpretation as <code>value1</code> .

Returns:

<code>options</code>	The updated optimization options structure.
----------------------	---

Examples:

```
% Obtain the default options.
opt = optimset

% Modify the value of the parameter
% display in the optimization
% options structure
opt = optimset(opt,'display','on');

% Return default options
opt = optimset('whatever')

% Modify a MOSEK parameter.
opt = [];
opt = optimset(opt,'MSK_DPAR_INTPNT_TOLMURED',1.0e-14);
```

Chapter 9

Case studies

9.1 Robust linear optimization

In most linear optimization examples discussed in this manual it is implicitly assumed that the problem data, such as c and A , is known with certainty. However, in practice this is seldom the case, e.g. the data may just be roughly estimated, affected by measurement errors or be affected by random events.

In this section a robust linear optimization methodology is presented which removes the assumption that the problem data is known exactly. Rather it is assumed that the data belongs to some set, i.e. a box or an ellipsoid.

The computations are performed using the MOSEK optimization toolbox for MATLAB but could equally well have been implemented using the MOSEK API.

This section is co-authored with A. Ben-Tal and A. Nemirovski.

9.1.1 Introductory example

Consider the following toy-sized linear optimization problem: A company produces two kinds of drugs, DrugI and DrugII, containing a specific active agent A, which is extracted from a raw materials that should be purchased on the market. The drug production data are as follows:

Drug	Selling price, \$ per 1000 packs	Content of agent A, gm per 1000 packs	Production expenses per 1000 packs		
			manpower, hours	equipment, hours	operational costs, \$
DrugI	6200	0.500	90.0	40.0	700
DrugII	6900	0.600	100.0	50.0	800

There are two kinds of raw materials, RawI and RawII, which can be used as sources of the active agent. The related data is as follows:

Raw material	Purchasing price, \$ per kg	Content of agent A, gm per kg
RawI	100.00	0.01
RawII	199.90	0.02

Finally, the monthly resources dedicated to producing the drugs are as follows:

Budget, \$	Manpower, hours	Equipment, hours	Capacity of raw materials storage, kg
100000	2000	800	1000

The problem is to find the production plan which maximizes the profit of the company, i.e. minimize the purchasing and operational costs

$$100 \cdot \text{RawI} + 199.90 \cdot \text{RawII} + 700 \cdot \text{DrugI} + 800 \cdot \text{DrugII}$$

and maximize the income

$$6200 \cdot \text{DrugI} + 6900 \cdot \text{DrugII}$$

The problem can be stated as the following linear programming program:

Minimize

$$-(100 \cdot \text{RawI} + 199.90 \cdot \text{RawII} + 700 \cdot \text{DrugI} + 800 \cdot \text{DrugII}) + (6200 \cdot \text{DrugI} + 6900 \cdot \text{DrugII}) \quad (9.1)$$

subject to

$$\begin{aligned}
0.01 \cdot \text{RawI} + 0.02 \cdot \text{RawII} - 0.500 \cdot \text{DrugI} - 0.600 \cdot \text{DrugII} &\geq 0 & (a) \\
\text{RawI} + \text{RawII} &\leq 1000 & (b) \\
90.0 \cdot \text{DrugI} + 100.0 \cdot \text{DrugII} &\leq 2000 & (c) \\
40.0 \cdot \text{DrugI} + 50.0 \cdot \text{DrugII} &\leq 800 & (d) \\
100.0 \cdot \text{RawI} + 199.90 \cdot \text{RawII} + 700 \cdot \text{DrugI} + 800 \cdot \text{DrugII} &\leq 100000 & (d) \\
\text{RawI}, \text{RawII}, \text{DrugI}, \text{DrugII} &\geq 0 & (e)
\end{aligned} \quad (9.2)$$

where the variables are the amounts RawI , RawII (in kg) of raw materials to be purchased and the amounts DrugI , DrugII (in 1000 of packs) of drugs to be produced. The objective (9.1) denotes the profit to be maximized, and the inequalities can be interpreted as follows:

- (a) Balance of the active agent.
- (b) Storage restriction.
- (c) Manpower restriction.
- (d) Equipment restriction.
- (e) Budget restriction.

Here is the MATLAB script which specifies the problem and solves it using the MOSEK optimization toolbox:

```
% rlo1.m

clear prob;

prob.c = [-100;-199.9;6200-700;6900-800];
prob.a = sparse([0.01,0.02,-0.500,-0.600;1,1,0,0;
                0,0,90.0,100.0;0,0,40.0,50.0;100.0,199.9,700,800]);
prob.blc = [0;-inf;-inf;-inf;-inf];
prob.buc = [inf;1000;2000;800;100000];
prob.blx = [0;0;0;0];
prob.bux = [inf;inf;inf;inf];
[r,res] = mosekopt('maximize',prob);
xx      = res.sol.itr.xx;
RawI    = xx(1);
RawII   = xx(2);
DrugI   = xx(3);
DrugII  = xx(4);

disp(sprintf('*** Optimal value: %8.3f',prob.c'*xx));
disp('*** Optimal solution:');
disp(sprintf('RawI:    %8.3f',RawI));
disp(sprintf('RawII:   %8.3f',RawII));
disp(sprintf('DrugI:   %8.3f',DrugI));
disp(sprintf('DrugII:  %8.3f',DrugII));
```

When executing this script, the following is displayed:

```
*** Optimal value: 8819.658
*** Optimal solution:
RawI:    0.000
RawII:   438.789
DrugI:   17.552
DrugII:  0.000
```

We see that the optimal solution promises the company a modest but quite respectful profit of 8.8%. Please note that at the optimal solution the balance constraint is active: the production process utilizes the full amount of the active agent contained in the raw materials.

9.1.2 Data uncertainty and its consequences.

Please note that not all problem data can be regarded as “absolutely reliable”; e.g. one can hardly believe that the contents of the active agent in the raw materials are *exactly* the “nominal data” 0.01 gm/kg for RawI and 0.02 gm/kg for RawII. In reality, these contents definitely vary around the indicated values. A natural assumption here is that the actual contents of the active agent a_I in RawI and a_{II} in RawII are realizations of random variables somehow distributed around the “nominal contents” $a_I^n = 0.01$ and $a_{II}^n = 0.02$. To be more

specific, assume that a_I drifts in the 0.5% margin of a_I^n , i.e. it takes with probability 0.5 the values from the interval $a_I^n(1 \pm 0.005) = a_I^n\{0.00995; 0.01005\}$. Similarly, assume that a_{II} drifts in the 2% margin of a_{II}^n , taking with probabilities 0.5 the values $a_{II}^n(1 \pm 0.02) = a_{II}^n\{0.0196; 0.0204\}$. How do the perturbations of the contents of the active agent affect the production process?

The optimal solution prescribes to purchase 438.8 kg of RawII and to produce 17552 packs of DrugI. With the above random fluctuations in the content of the active agent in RawII, this production plan, with probability 0.5, will be infeasible – with this probability, the actual content of the active agent in the raw materials will be less than required to produce the planned amount of DrugI. For the sake of simplicity, assume that this difficulty is resolved in the simplest way: when the actual content of the active agent in the raw materials is insufficient, the output of the drug is reduced accordingly. With this policy, the actual production of DrugI becomes a random variable which takes, with probabilities 0.5, the nominal value of 17552 packs and the 2% less value of 17201 packs. These 2% fluctuations in the production affect the profit as well; the latter becomes a random variable taking, with probabilities 0.5, the nominal value 8,820 and the 21% less value 6,929. The expected profit is 7,843, which is by 11% less than the nominal profit 8,820 promised by the optimal solution of the problem.

We see that in our toy example that small (and in reality unavoidable) perturbations of the data may make the optimal solution infeasible, and a straightforward adjustment to the actual solution values may heavily affect the solution quality.

It turns out that the outlined phenomenon is found in many linear programs of practical origin. Usually, in these programs at least part of the data is not known exactly and can vary around its nominal values, and these data perturbations can make the nominal optimal solution – the one corresponding to the nominal data – infeasible. It turns out that the consequences of data uncertainty can be much more severe than in our toy example. The analysis of linear optimization problems from the NETLIB collection¹ reported in [13] demonstrates that for 13 of 94 NETLIB problems, already 0.01% perturbations of “clearly uncertain” data can make the nominal optimal solution severely infeasible: with these perturbations, the solution, with a non-negligible probability, violates some of the constraints by 50% and more. It should be added that in the general case, in contrast to the toy example we have considered, there is no evident way to adjust the optimal solution by a small modification to the actual values of the data. Moreover there are cases when such an adjustment is impossible — in order to become feasible for the perturbed data, the nominal optimal solution should be “completely reshaped”.

¹NETLIB is a collection of LP's, mainly of the real world origin, which is a standard benchmark for evaluating LP algorithms

9.1.3 Robust linear optimization methodology

A natural approach to handling data uncertainty in optimization is offered by the *Robust Optimization Methodology* which, as applied to linear optimization, is as follows.

9.1.3.1 Uncertain linear programs and their robust counterparts.

Consider a linear optimization problem

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && l_c \leq Ax \leq u_c, \\ & && l_x \leq x \leq u_x, \end{aligned} \tag{9.3}$$

with the data $(c, A, l_c, u_c, l_x, u_x)$, and assume that this data is not known exactly; all we know is that the data varies in a given *uncertainty set* \mathcal{U} . The simplest example is the one of *interval uncertainty*, where every data entry can run through a given interval:

$$\begin{aligned} \mathcal{U} = & \left\{ (c, A, l_c, u_c, l_x, u_x) : \right. \\ & (c^n - dc, A^n - dA, l_c^n - dl_c, u_c^n - du_c, l_x^n - dl_x, u_x^n - du_x) \leq (c, A, l_c, u_c, l_x, u_x) \\ & \left. \leq (c^n + dc, A^n + dA, l_c^n + dl_c, u_c^n + du_c, l_x^n + dl_x, u_x^n + du_x) \right\}. \end{aligned} \tag{9.4}$$

Here

$$(c^n, A^n, l_c^n, u_c^n, l_x^n, u_x^n)$$

is the *nominal data*,

$$dc, dA, dl_c, du_c, dl_x, du_x \geq 0$$

is the *data perturbation bounds*. Please note that some of the entries in the data perturbation bounds can be zero, meaning that the corresponding data entries are certain (the expected values equals the actual values).

- The family of instances (9.3) with data running through a given uncertainty set \mathcal{U} is called an *uncertain linear optimization problem*.
- A vector x is called a *robust feasible solution* to an uncertain linear optimization problem, if it remains feasible for all realizations of the data from the uncertainty set, i.e. if

$$l_c \leq Ax \leq u_c, l_x \leq x \leq u_x \text{ for all } (c, A, l_c, u_c, l_x, u_x) \in \mathcal{U}. \tag{9.5}$$

- If for some value t we have $c^T x \leq t$ for all realizations of the objective from the uncertainty set, we say that *robust value of the objective* at x does not exceed t .

The Robust Optimization methodology proposes to associate with an uncertain linear program its *robust counterpart* (RC) which is *the problem of minimizing the robust optimal value over the set of all robust feasible solutions*, i.e. the problem

$$\min_{t,x} \{t : c^T x \leq t, l_c \leq Ax \leq u_c, l_x \leq x \leq u_x \text{ for all } (c, A, l_c, u_c, l_x, u_x) \in \mathcal{U}\}. \quad (9.6)$$

The optimal solution to (9.6) is treated as the “uncertainty-immuned” solution to the original uncertain linear programming program.

9.1.3.2 Robust counterpart of an uncertain linear optimization problem with interval uncertainty

In general, the RC (9.6) of an uncertain linear optimization problem is not a linear optimization problem since (9.6) has infinitely many linear constraints. There are, however, cases when (9.6) can be rewritten equivalently as a linear programming program; in particular, this is the case for interval uncertainty (9.4). Specifically, in the case of (9.4), the robust counterpart of uncertain linear program is equivalent to the following linear program in variables x, y, t :

$$\begin{array}{llll} \text{minimize} & t & & \\ \text{subject to} & (c^n)^T x + (dc)^T y - t \leq 0, & (a) & \\ & l_c^n + dl_c \leq (A^n)x - (dA)y, & (b) & \\ & (A^n)x + (dA)y \leq u_c^n - du_c, & (c) & \\ & 0 \leq x + y, & (d) & \\ & 0 \leq -x + y, & (e) & \\ & l_x^n + dl_x \leq x \leq u_x^n - du_x, & (f) & \end{array} \quad (9.7)$$

The origin of (9.7) is quite transparent: The constraints (9.7.d – e) linking x and y merely say that $y_i \geq |x_i|$ for all i . With this in mind, it is evident that at every feasible solution to (9.7) the entries in the vector

$$(A^n)x - (dA)y$$

are lower bounds on the entries of Ax with A from the uncertainty set (9.4), so that (9.7.b) ensures that $l_c \leq Ax$ for all data from the uncertainty set. Similarly, (9.7.c) and (9.7.a), (9.7.f) ensure, for all data from the uncertainty set, that $Ax \leq u_c$, $c^T x \leq t$, and that the entries in x satisfy the required lower and upper bounds, respectively.

Please note that at the optimal solution to (9.7), one clearly has $y_j = |x_j|$. It follows that when the bounds on the entries of x impose nonnegativity (nonpositivity) of an entry x_j , then there is no need to introduce the corresponding additional variable y_i — from the very beginning it can be replaced with x_j , if x_j is nonnegative, or with $-x_j$, if x_j is nonpositive.

Another possible formulation of problem (9.7) is the following. Let

$$l_c^n + dl_c = (A^n)x - (dA)y - f, \quad f \geq 0$$

then this equation is equivalent to (a) in (9.7.b). If $(l_c)_i = -\infty$, then equation i should be dropped from the computations. Similarly,

$$-x + y = g \geq 0$$

is equivalent to (9.7.d). This implies that

$$l_c^n + dl_c - (A^n)x + f = -(dA)y$$

and that

$$y = g + x$$

Substituting these values into (9.7) gives

$$\begin{array}{llll} \text{minimize} & t & & \\ \text{subject to} & (c^n)^T x + (dc)^T (g + x) - t & \leq & 0, \\ & 0 & \leq & f, \\ & 2(A^n)x + (dA)(g + x) + f + l_c^n + dl_c & \leq & u_c^n - du_c, \\ & 0 & \leq & g, \\ & 0 & \leq & 2x + g, \\ & l_x^n + dl_x & \leq & x \leq u_x^n - du_x, \end{array} \quad (9.8)$$

which after some simplifications leads to

$$\begin{array}{llll} \text{minimize} & t & & \\ \text{subject to} & (c^n + dc)^T x + (dc)^T g - t & \leq & 0, \quad (a) \\ & 0 & \leq & f, \quad (b) \\ & 2(A^n + dA)x + (dA)g + f - (l_c^n + dl_c) & \leq & u_c^n - du_c, \quad (c) \\ & 0 & \leq & g, \quad (d) \\ & 0 & \leq & 2x + g, \quad (e) \\ & l_x^n + dl_x & \leq & x \leq u_x^n - du_x, \quad (f) \end{array} \quad (9.9)$$

and

$$\begin{array}{llll} \text{minimize} & t & & \\ \text{subject to} & (c^n + dc)^T x + (dc)^T g - t & \leq & 0, \quad (a) \\ & 2(A^n + dA)x + (dA)g + f & \leq & u_c^n - du_c + l_c^n + dl_c, \quad (b) \\ & 0 & \leq & 2x + g, \quad (c) \\ & 0 & \leq & f, \quad (d) \\ & 0 & \leq & g, \quad (e) \\ & l_x^n + dl_x & \leq & x \leq u_x^n - du_x. \quad (f) \end{array} \quad (9.10)$$

Please note that this problem has more variables but much fewer constraints than (9.7). Therefore, (9.10) is likely to be solved faster than (9.7). Note too that (9.10.b) is trivially redundant if $l_x^n + dl_x \geq 0$.

9.1.3.3 Introductory example (continued)

Let us apply the Robust Optimization methodology to our drug production example presented in Section 9.1.1, assuming that the only uncertain data is the contents of the active agent in the raw materials, and that these contents vary in 0.5% and 2% neighborhoods of the respective nominal values 0.01 and 0.02. With this assumption, the problem becomes an uncertain LP affected by interval uncertainty; the robust counterpart (9.7) of this uncertain LP is the linear program

$$\begin{aligned}
 & \text{(Drug_RC) :} \\
 & \text{maximize} \\
 & \qquad \qquad \qquad t \\
 & \text{subject to} \\
 & \quad t \leq -100 \cdot \text{RawI} - 199.9 \cdot \text{RawII} + 5500 \cdot \text{DrugI} + 6100 \cdot \text{DrugII} \\
 & \quad 0.01 \cdot 0.995 \cdot \text{RawI} + 0.02 \cdot 0.98 \cdot \text{RawII} - 0.500 \cdot \text{DrugI} - 0.600 \cdot \text{DrugII} \geq 0 \\
 & \qquad \qquad \qquad \text{RawI} + \text{RawII} \leq 1000 \\
 & \qquad \qquad \qquad 90.0 \cdot \text{DrugI} + 100.0 \cdot \text{DrugII} \leq 2000 \\
 & \qquad \qquad \qquad 40.0 \cdot \text{DrugI} + 50.0 \cdot \text{DrugII} \leq 800 \\
 & \quad 100.0 \cdot \text{RawI} + 199.90 \cdot \text{RawII} + 700 \cdot \text{DrugI} + 800 \cdot \text{DrugII} \leq 100000 \\
 & \qquad \qquad \qquad \text{RawI}, \text{RawII}, \text{DrugI}, \text{DrugII} \geq 0
 \end{aligned}$$

Solving this problem with MOSEK we get the following output:

```

*** Optimal value: 8294.567
*** Optimal solution:
RawI:      877.732
RawII:      0.000
DrugI:      17.467
DrugII:      0.000

```

We see that the robust optimal solution we have built “costs money” – it promises a profit of just \$ 8,295 (cf. with the profit of \$ 8,820 promised by the nominal optimal solution). Please note, however, that the robust optimal solution remains feasible whatever are the realizations of the uncertain data from the uncertainty set in question, while the nominal optimal solution requires adjustment to this data and, with this adjustment, results in the average profit of \$ 7,843, which is by 5.4% *less* than the profit of \$ 8,295 *guaranteed* by the robust optimal solution. Note too that the robust optimal solution is significantly different from the nominal one: both solutions prescribe to produce the same drug DrugI (in the amounts 17,467 and 17,552 packs, respectively) but from different raw materials, RawI in the case of the robust solution and RawII in the case of the nominal solution. The reason is that although the price per unit of the active agent for RawII is slightly less than for RawI, the content of the agent in RawI is more stable, so when possible fluctuations of the contents are taken into account, RawI turns out to be more profitable than RawII.

9.1.4 Random uncertainty and ellipsoidal robust counterpart

In some cases, it is natural to assume that the perturbations affecting different uncertain data entries are random and independent of each other. In these cases, the robust counterpart based on the interval model of uncertainty seems to be too conservative: Why should we expect that all the data will be simultaneously driven to its most unfavorable values and immune the solution against this highly unlikely situation? A less conservative approach is offered by the *ellipsoidal* model of uncertainty. To motivate this model, let us see what happens with a particular linear constraint

$$a^T x \leq b \quad (9.11)$$

at a given candidate solution x in the case when the vector a of coefficients of the constraint is affected by random perturbations:

$$a = a^n + \zeta, \quad (9.12)$$

where a^n is the vector of nominal coefficients and ζ is a random perturbation vector with zero mean and covariance matrix V_a . In this case the value of the left-hand side of (9.11), evaluated at a given x , becomes a random variable with the expected value $(a^n)^T x$ and the standard deviation $\sqrt{x^T V_a x}$. Now let us act as an engineer who believes that the value of a random variable never exceeds its mean plus 3 times the standard deviation; we do not intend to be that specific and replace “3” in the above rule by a safety parameter Ω which will be in our control. Believing that the value of a random variable “never” exceeds its mean plus Ω times the standard deviation, we conclude that a “safe” version of (9.11) is the inequality

$$(a^n)^T x + \Omega \sqrt{x^T V_a x} \leq b. \quad (9.13)$$

The word “safe” above admits a quantitative interpretation: If x satisfies (9.13), one can bound from above the probability of the event that random perturbations (9.12) result in violating the constraint (9.11) evaluated at x . The bound in question depends on what we know about the distribution of ζ , e.g.

1. We always have the bound given by the Tschebyshev inequality:

$$\text{satisfies (9.13)} \Rightarrow \text{Prob} \{a^T x > b\} \leq \frac{1}{\Omega^2}. \quad (9.14)$$

2. When ζ is Gaussian, then the Tschebyshev bound can be improved to

$$\text{satisfies (9.13)} \Rightarrow \text{Prob} \{a^T x > b\} \leq \frac{1}{\sqrt{2\pi}} \int_{\Omega}^{\infty} \exp\{-t^2/2\} dt \leq 0.5 \exp\{-\Omega^2/2\}. \quad (9.15)$$

3. Assume that $\zeta = D\xi$, where Δ is certain $n \times m$ matrix, and $\xi = (\xi_1, \dots, \xi_m)^T$ is a random vector with independent coordinates ξ_1, \dots, ξ_m symmetrically distributed in the segment $[-1, 1]$. Setting $V = DD^T$ (V is a natural “upper bound” on the covariance matrix of ζ), one has

$$x \text{ satisfies (9.13)} \Rightarrow \text{Prob}\{a^T x > b\} \leq 0.5 \exp\{-\Omega^2/2\}. \quad (9.16)$$

Please note that in order to ensure the bounds in (9.15) and (9.16) to be $\leq 10^{-6}$, it suffices to set $\Omega = 5.13$.

Now, assume that we are given a linear program affected by random perturbations:

$$\begin{aligned} & \text{minimize} && [c^n + dc]^T x \\ & \text{subject to} && (l_c)_i \leq [a_i^n + da_i]^T x \leq (u_c)_i, \quad i = 1, \dots, m, \\ & && l_x \leq x \leq u_x, \end{aligned} \quad (9.17)$$

where $(c^n, \{a_i^n\}_{i=1}^m, l_c, u_c, l_x, u_x)$ are the nominal data, and dc, da_i are random perturbations with zero means². Assume, for the sake of definiteness, that every one of the random perturbations dc, da_1, \dots, da_m satisfies either the assumption of item 2 or the assumption of item 3, and let V_c, V_1, \dots, V_m be the corresponding (upper bounds on the) covariance matrices of the perturbations. Choosing a safety parameter Ω and replacing the objective and the bodies of all the constraints by their safe bounds as explained above, we arrive at the following optimization problem:

$$\begin{aligned} & \text{minimize} && t \\ & \text{subject to} && [c^n]^T x + \Omega \sqrt{x^T V_c x} \leq t, \\ & && (l_c)_i \leq [a_i^n]^T x - \Omega \sqrt{x^T V_{a_i} x}, \\ & && [a_i^n]^T x + \Omega \sqrt{x^T V_{a_i} x} \leq (u_c)_i, \quad i = 1, \dots, m, \\ & && l_x \leq x \leq u_x. \end{aligned} \quad (9.18)$$

The relation between problems (9.18) and (9.17) is as follows:

If (x, t) is a feasible solution of (9.18), then with probability at least

$$p = 1 - (m+1) \exp\{-\Omega^2/2\}$$

x is feasible for randomly perturbed problem (9.17), and t is an upper bound on the objective of (9.17) evaluated at x .

We see that if Ω is not too small (9.18) can be treated as a “safe version” of (9.17).

On the other hand, it is easily seen that (9.18) is nothing but the robust counterpart of the uncertain linear optimization problem with the nominal data $(c^n, \{a_i^n\}_{i=1}^m, l_c, u_c, l_x, u_x)$ and the

²For the sake of simplicity, we assume that the bounds l_c, u_c, l_x, u_x are not affected by uncertainty; extensions to the case when it is not so are evident.

row-wise ellipsoidal uncertainty given by the matrices $V_c, V_{a_1}, \dots, V_{a_m}$. In the corresponding uncertainty set, the uncertainty affects the coefficients of the objective and the constraint matrix only, and the perturbation vectors affecting the objective and the vectors of coefficients of the linear constraints run, independently of each other, through the respective ellipsoids

$$\begin{aligned} E_c &= \left\{ dc = \Omega V_c^{1/2} u : u^T u \leq 1 \right\}, \\ E_{a_i} &= \left\{ da_i = \Omega V_{a_i}^{1/2} u : u^T u \leq 1 \right\}, i = 1, \dots, m. \end{aligned}$$

It turns out that in many cases the ellipsoidal model of uncertainty is significantly less conservative and thus better suited for practice, than the interval model of uncertainty.

Last but not least, it should be mentioned that problem (9.18) is equivalent to a conic quadratic program, specifically to the program

$$\begin{aligned} &\text{minimize} && t \\ &\text{subject to} && [c^n]^T x + \Omega z \leq t, \\ & && (l_c)_i \leq [a_i^n]^T x - \Omega z_i, \\ & && [a_i^n]^T x + \Omega z_i \leq (u_c)_i, \quad i = 1, \dots, m, \\ & && 0 = w - D_c x \\ & && 0 = w^i - D_{a_i} x, \quad i = 1, \dots, m, \\ & && 0 \leq z - \sqrt{w^T w}, \\ & && 0 \leq z_i - \sqrt{(w^i)^T w^i}, \quad i = 1, \dots, m, \\ & && l_x \leq x \leq u_x. \end{aligned} \tag{9.19}$$

where D_c and D_{a_i} are matrices satisfying the relations

$$V_c = D_c^T D_c, \quad V_{a_i} = D_{a_i}^T D_{a_i}, \quad i = 1, \dots, m.$$

9.1.4.1 Example: Interval and Ellipsoidal robust counterparts of uncertain linear constraint with independent random perturbations of coefficients

Consider a linear constraint

$$l \leq \sum_{j=1}^n a_j x_j \leq u \tag{9.20}$$

and assume that the a_j coefficients of the body of the constraint are uncertain and vary in intervals $a_j^n \pm \sigma_j$. The worst-case-oriented model of uncertainty here is the interval one, and the corresponding robust counterpart of the constraint is given by the system of linear inequalities

$$\begin{aligned} l &\leq \sum_{j=1}^n a_j^n x_j - \sum_{j=1}^n \sigma_j y_j, \\ &\sum_{j=1}^n a_j^n x_j + \sum_{j=1}^n \sigma_j y_j \leq u, \\ 0 &\leq x_j + y_j, \\ 0 &\leq -x_j + y_j, \quad j = 1, \dots, n. \end{aligned} \tag{9.21}$$

Now, assume that we have reasons to believe that the true values of the coefficients a_j are obtained from their nominal values a_j^n by random perturbations, independent for different j and symmetrically distributed in the segments $[-\sigma_j, \sigma_j]$. With this assumption, we are in the situation of item 3 and can replace the uncertain constraint (9.20) with its ellipsoidal robust counterpart

$$\begin{aligned} l &\leq \sum_{j=1}^n a_j^n x_j - \Omega z, \\ \sum_{j=1}^n a_j^n x_j + \Omega z &\leq u, \\ 0 &\leq z - \sqrt{\sum_{j=1}^n \sigma_j^2 x_j^2}. \end{aligned} \tag{9.22}$$

Please note that with the model of random perturbations, a vector x satisfying (9.22) satisfies a realization of (9.20) with probability at least $1 - \exp\{\Omega^2/2\}$; for $\Omega = 6$. This probability is $\geq 1 - 1.5 \cdot 10^{-8}$, which for all practical purposes is the same as saying that x satisfies all realizations of (9.20). On the other hand, the uncertainty set associated with (9.21) is the box

$$B = \{a = (a_1, \dots, a_n)^T : a_j^n - \sigma_j \leq a_j \leq a_j^n + \sigma_j, j = 1, \dots, n\},$$

while the uncertainty set associated with (9.22) is the ellipsoid

$$E(\Omega) = \left\{ a = (a_1, \dots, a_n)^T : \sum_{j=1}^n (a_j - a_j^n)^2 \sigma_j^2 \leq \Omega^2 \right\}.$$

For a moderate value of Ω , say $\Omega = 6$, and $n \geq 40$, the ellipsoid $E(\Omega)$ in its diameter, typical linear sizes, volume, etc. is incomparably less than the box B , the difference becoming more dramatic the larger the dimension n of the box and the ellipsoid. It follows that the ellipsoidal robust counterpart (9.22) of the randomly perturbed uncertain constraint (9.20) is much less conservative than the interval robust counterpart (9.21), while ensuring basically the same “robustness guarantees”. To illustrate this important point, consider the following numerical examples:

There are n different assets on the market. The return on \$ 1 invested in asset j is a random variable distributed symmetrically in the segment $[\delta_j - \sigma_j, \delta_j + \sigma_j]$, and the returns on different assets are independent of each other. The problem is to distribute \$ 1 among the assets in order to get the largest possible total return on the resulting portfolio.

A natural model of the problem is an uncertain linear optimization problem

$$\begin{aligned}
 & \text{maximize} && \sum_{j=1}^n a_j x_j \\
 & \text{subject to} && \sum_{j=1}^n x_j = 1, \\
 & && 0 \leq x_j, \quad j = 1, \dots, n.
 \end{aligned} \tag{9.23}$$

where a_j are the uncertain returns of the assets. Both the nominal optimal solution (set all returns a_j equal to their nominal values δ_j) and the risk-neutral Stochastic Programming approach (maximize the expected total return) result in the same solution: Our \$ 1 should be invested in the most promising asset(s) – the one(s) with the maximal nominal return. This solution, however, can be very unreliable if, as is typically the case in reality, the most promising asset has the largest volatility σ and is in this sense the most risky. To reduce the risk, one can use the Robust Counterpart approach which results in the following optimization problems.

The Interval Model of Uncertainty:

$$\begin{aligned}
 & \text{maximize} && t \\
 & \text{subject to} && 0 \leq -t + \sum_{j=1}^n (\delta_j - \sigma_j) x_j, \\
 & && \sum_{j=1}^n x_j = 1, \\
 & && 0 \leq x_j, \quad j = 1, \dots, n
 \end{aligned} \tag{9.24}$$

and

The ellipsoidal Model of Uncertainty:

$$\begin{aligned}
 & \text{maximize} && t \\
 & \text{subject to} && 0 \leq -t + \sum_{j=1}^n (\delta_j) x_j - \Omega z, \\
 & && 0 \leq z - \sqrt{\sum_{j=1}^n \sigma_j^2 x_j^2}, \\
 & && \sum_{j=1}^n x_j = 1, \\
 & && 0 \leq x_j, \quad j = 1, \dots, n.
 \end{aligned} \tag{9.25}$$

Note that the problem (9.25) is essentially the risk-averted portfolio model proposed in mid-50's by Markowitz.

The solution of (9.24) is evident — our \$ 1 should be invested in the asset(s) with the largest possible *guaranteed* return $\delta_j - \sigma_j$. In contrast to this very conservative policy (which in reality prescribes to keep the initial capital in a bank or in the most reliable, and thus low profit, assets), the optimal solution to (9.25) prescribes a quite reasonable diversification of

investments which allows to get much better total return than (9.24) with basically zero risk³. To illustrate this, assume that there are $n = 300$ assets with the nominal returns (per year) varying from 1.04 (bank savings) to 2.00:

$$\delta_j = 1.04 + 0.96 \frac{j-1}{n-1}, j = 1, 2, \dots, n = 300$$

and volatilities varying from 0 for the bank savings to 1.2 for the most promising asset:

$$\sigma_j = 1.152 \frac{j-1}{n-1}, j = 1, \dots, n = 300.$$

Here is a MATLAB script which builds the associated problem (9.25), solves it via the MOSEK optimization toolbox, displays the resulting robust optimal value of the total return and the distribution of investments, and finally runs 10,000 simulations to get the distribution of the total return on the resulting portfolio (in these simulations, the returns on all assets are uniformly distributed in the corresponding intervals):

```
% File: rlo2.m

% Problem:
%
% Maximize t subject to
% t <= sum(delta(j)*x(j)) -Omega*z,
% y(j) = sigma(j)*x(j), j=1,...,n,
% sum(x(j)) = 1,
% norm(y) <= z,
% 0 <= x.

clear prob;
n      = 300;
Omega = 6;

% Set nominal returns and volatilities
delta = (0.96/(n-1))*[0:1:n-1]+1.04;
sigma = (1.152/(n-1))*[0:1:n-1];

% Set mosekopt description of the problem
prob.c = -[1;zeros(2*n+1,1)];
A       = [-1,ones(1,n)+delta,-Omega,zeros(1,n);zeros(n+1,2*n+2)];
for j=1:n,
    % Body of the constraint y(j) - sigma(j)*x(j) = 0:
    A(j+1,j+1) = -sigma(j);
    A(j+1,2+n+j) = 1;
end;
```

³Recall that in our discussion we have assumed the returns on different assets to be independent of each other. In reality, this is not so and this is why diversification of investments, although reducing the risk, never eliminates it completely

```

A(n+2,2:n+1)      = ones(1,n);
prob.a             = sparse(A);
prob.blc           = [zeros(n+1,1);1];
prob.buc           = [inf;zeros(n,1);1];
prob.blx           = [-inf;zeros(n,1);0;zeros(n,1)];
prob.bux           = inf*ones(2*n+2,1);
prob.cones         = cell(1,1);
prob.cones{1}.type = 'MSK_CT_QUAD';
prob.cones{1}.sub  = [n+2;[n+3:1:2*n+2]'];

% Run mosekopt
[r,res]=mosekopt('minimize echo(1)',prob);

% Display the solution
xx = res.sol.itr.xx;
t  = xx(1);

disp(sprintf('Robust optimal value: %5.4f',t));
x = max(xx(2:1+n),zeros(n,1));
plot([1:1:n],x,'-m');
grid on;

disp('Press <Enter> to run simulations');
pause

% Run simulations

Nsim = 10000;
out  = zeros(Nsim,1);
for i=1:Nsim,
    returns = delta+(2*rand(1,n)-1).*sigma;
    out(i)  = returns*x;
end;
disp(sprintf('Actual returns over %d simulations:',Nsim));
disp(sprintf('Min=%5.4f Mean=%5.4f Max=%5.4f StD=%5.2f',...
    min(out),mean(out),max(out),std(out)));
hist(out);

```

Here are the results displayed by the script:

```

Robust optimal value: 1.3428
Actual returns over 10000 simulations:
Min=1.5724 Mean=1.6965 Max=1.8245 StD= 0.03

```

Please note that with our set-up there is exactly one asset with guaranteed return greater than 1 – asset # 1 (bank savings, return 1.04, zero volatility). Consequently, the interval robust counterpart (9.24) prescribes to put our \$ 1 in the bank, thus getting a 4% profit. In contrast to this, the diversified portfolio given by the optimal solution of (9.25) never yields profit less than 57.2%, and yields at average a 69.67% profit with pretty low (0.03) standard

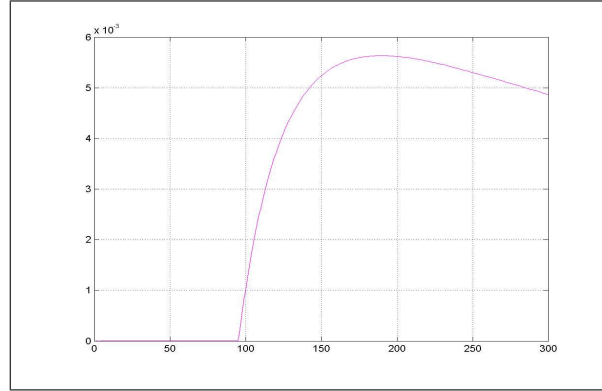


Figure 9.1: Distribution of investments among the assets in the optimal solution of.

deviation. We see that in favorable circumstances the ellipsoidal robust counterpart of an uncertain linear program indeed is less conservative than, although basically as reliable as, the interval robust counterpart.

Finally, let us compare our results with those given by the nominal optimal solution. The latter prescribes to invest everything we have in the most promising asset (in our example this is the asset # 300 with a nominal return of 2.00 and volatility of 1.152). Assuming that the actual return is uniformly distributed in the corresponding interval and running 10,000 simulations, we get the following results:

Nominal optimal value: 2.0000

Actual returns over 10000 simulations:

Min=0.8483 Mean=1.9918 Max=3.1519 StD= 0.66

We see that the nominal solution results in a portfolio which is much more risky, although better at average, than the portfolio given by the robust solution.

9.1.4.2 Combined Interval-Ellipsoidal Robust Counterpart

We have considered the case when the coefficients a_j of uncertain linear constraint (9.20) are affected by uncorrelated random perturbations symmetrically distributed in given intervals $[-\sigma_j, \sigma_j]$, and we have discussed two ways to model the uncertainty:

- The interval uncertainty model (the uncertainty set \mathcal{U} is the box B), where we ignore the stochastic nature of the perturbations and their independence. This model yields the Interval Robust Counterpart (9.21);
- The ellipsoidal uncertainty model (\mathcal{U} is the ellipsoid $E(\Omega)$), which takes into account the stochastic nature of data perturbations and yields the Ellipsoidal Robust Counterpart (9.22).

Please note that although for large n the ellipsoid $E(\Omega)$ in its diameter, volume and average linear sizes is incomparably smaller than the box B , in the case of $\Omega > 1$ the ellipsoid $E(\Omega)$ in certain directions goes beyond the box. E.g. the ellipsoid $E(6)$, although much more narrow than B in most of the directions, is 6 times wider than B in the directions of the coordinate axes. Intuition says that it hardly makes sense to keep in the uncertainty set realizations of the data which are outside of B and thus forbidden by our model of perturbations, so in the situation under consideration the intersection of $E(\Omega)$ and B is a better model of the uncertainty set than the ellipsoid $E(\Omega)$ itself. What happens when the model of the uncertainty set is the “combined interval-ellipsoidal” uncertainty $\mathcal{U}(\Omega) = E(\Omega) \cap B$?

First, it turns out that the RC of (9.20) corresponding to the uncertainty set $\mathcal{U}(\Omega)$ is still given by a system of linear and conic quadratic inequalities, specifically the system

$$\begin{aligned}
 l &\leq \sum_{j=1}^n a_j^n x_j - \sum_{j=1}^n \sigma_j y_j - \Omega \sqrt{\sum_{j=1}^n \sigma_j^2 u_j^2}, \\
 \sum_{j=1}^n a_j^n x_j + \sum_{j=1}^n \sigma_j z_j + \Omega \sqrt{\sum_{j=1}^n \sigma_j^2 v_j^2} &\leq u, \\
 -y_j &\leq x_j - u_j &\leq y_j, \quad j = 1, \dots, n, \\
 -z_j &\leq x_j - v_j &\leq z_j, \quad j = 1, \dots, n.
 \end{aligned} \tag{9.26}$$

Second, it turns out that our intuition is correct: As a model of uncertainty, $\mathcal{U}(\Omega)$ is as reliable as the ellipsoid $E(\Omega)$. Specifically, if x can be extended to a feasible solution of (9.26), then the probability for x to satisfy a realization of (9.20) is $\geq 1 - \exp\{-\Omega^2/2\}$.

The conclusion is that if we have reasons to assume that the perturbations of uncertain coefficients in a constraint of an uncertain linear optimization problem are (a) random, (b) independent of each other, and (c) symmetrically distributed in given intervals, then it makes sense to associate with this constraint an interval-ellipsoidal model of uncertainty and use a system of linear and conic quadratic inequalities (9.26). Please note that when building the robust counterpart of an uncertain linear optimization problem, one can use different models of the uncertainty (e.g., interval, ellipsoidal, combined interval-ellipsoidal) for different uncertain constraints within the same problem.

9.1.5 Further references

For further information about robust linear optimization consult [13, 14].

9.2 Geometric (posynomial) optimization

9.2.1 The problem

A *geometric optimization* problem can be stated as follows

$$\begin{aligned} & \text{minimize} && \sum_{k \in J_0} c_k \prod_{j=0}^{n-1} t_j^{a_{kj}} \\ & \text{subject to} && \sum_{k \in J_i} c_k \prod_{j=0}^{n-1} t_j^{a_{kj}} \leq 1, \quad i = 1, \dots, m, \\ & && t > 0, \end{aligned} \tag{9.27}$$

where it is assumed that

$$\cup_{k=0}^m J_k = \{1, \dots, T\}$$

and if $i \neq j$, then

$$J_i \cap J_j = \emptyset.$$

Hence, A is a $T \times n$ matrix and c is a vector of length T . Given $c_k > 0$ then

$$c_k \prod_{j=0}^{n-1} t_j^{a_{kj}}$$

is called a *monomial*. A sum of monomials i.e.

$$\sum_{k \in J_i} c_k \prod_{j=0}^{n-1} t_j^{a_{kj}}$$

is called a *posynomial*. In general, the problem (9.27) is very hard to solve. However, the posynomial case where it is required that

$$c > 0$$

is relatively easy. The reason is that using a simple variable transformation a convex optimization problem can be obtained. Indeed using the variable transformation

$$t_j = e^{x_j} \tag{9.28}$$

we obtain the problem

$$\begin{aligned} & \text{minimize} && \sum_{k \in J_0} c_k e^{\sum_{j=0}^{n-1} a_{kj} x_j} \\ & \text{subject to} && \sum_{k \in J_i} c_k e^{\sum_{j=0}^{n-1} a_{kj} x_j} \leq 1, \quad i = 1, \dots, m, \end{aligned} \tag{9.29}$$

which is a convex optimization problem that can be solved using MOSEK. We will call

$$c_t e^{\left(\sum_{j=0}^{n-1} a_{tj} x_j\right)} = e^{\left(\log(c_t) + \sum_{j=0}^{n-1} a_{tj} x_j\right)}$$

a *term* and hence the number of terms is T .

As stated, the problem (9.29) is non-separable. However, using

$$v_t = \log(c_t) + \sum_{j=0}^{n-1} a_{tj} x_j$$

we obtain the separable problem

$$\begin{aligned} & \text{minimize} && \sum_{t \in J_0} e^{v_t} \\ & \text{subject to} && \sum_{t \in J_i} e^{v_t} \leq 1, && i = 1, \dots, m, \\ & && \sum_{j=0}^{n-1} a_{tj} x_j - v_t = -\log(c_t), && t = 0, \dots, T, \end{aligned} \tag{9.30}$$

which is a separable convex optimization problem.

A warning about this approach is that the exponential function e^x is only numerically well-defined for values of x in a small interval around 0 since e^x grows very rapidly as x becomes larger. Therefore numerical problems may arise when solving the problem on this form.

9.2.2 Applications

A large number of practical applications, particularly in electrical circuit design, can be cast as a geometric optimization problem. We will not review these applications here but rather refer the reader to [15] and the references therein.

9.2.3 Modeling tricks

A lot of tricks that can be used for modeling posynomial optimization problems are described in [15]. Therefore, in this section we cover only one important case.

9.2.3.1 Equalities

In general, equalities are not allowed in (9.27), i.e.

$$\sum_{k \in J_i} c_k \prod_{j=0}^{n-1} t_j^{a_{kj}} = 1$$

is not allowed. However, a monomial equality is not a problem. Indeed consider the example

$$xyz^{-1} = 1$$

of a monomial equality. The equality is identical to

$$1 \leq xyz^{-1} \leq 1$$

which in turn is identical to the two inequalities

$$\begin{aligned} xyz^{-1} &\leq 1, \\ \frac{1}{xyz^{-1}} &= x^{-1}y^{-1}z \leq 1. \end{aligned}$$

Hence, it is possible to model a monomial equality using two inequalities.

9.2.4 Problematic formulations

Certain formulations of geometric optimization problems may cause problems for the algorithms implemented in MOSEK. Basically there are two kinds of problems that may occur:

- The solution vector is finite, but an optimal objective value can only be approximated.
- The optimal objective value is finite but implies that a variable in the solution is infinite.

9.2.4.1 Finite unattainable solution

The following problem illustrates an unattainable solution:

$$\begin{aligned} &\text{minimize} && x^2y \\ &\text{subject to} && xy \leq 1, \\ &&& x, y > 0. \end{aligned}$$

Clearly, the optimal objective value is 0 but because of the constraint the $x, y > 0$ constraint this value can never be attained: To see why this is a problem, remember that MOSEK substitutes $x = e^{t_x}$ and $y = e^{t_y}$ and solves the problem as

$$\begin{aligned} &\text{minimize} && e^{2t_x}e^{t_y} \\ &\text{subject to} && e^{t_x}e^{t_y} \leq 1, \\ &&& t_x, t_y \in \mathbb{R}. \end{aligned}$$

The optimal solution implies that $t_x = -\infty$ or $t_y = -\infty$, and thus it is unattainable.

Now, the issue should be clear: If a variable x appears only with nonnegative exponents, then fixing $x = 0$ will minimize all terms in which it appears — but such a solution cannot be attained.

9.2.4.2 Infinite solution

A similar problem will occur if a finite optimal objective value requires a variable to be infinite. This can be illustrated by the following example:

$$\begin{array}{ll} \text{minimize} & x^{-2} \\ \text{subject to} & x^{-1} \leq 1, \\ & x > 0, \end{array}$$

which is a valid geometric programming problem. In this case the optimal objective is 0, but this requires $x = \infty$, which is unattainable.

Again, this specific case will appear if a variable x appears only with negative exponents in the problem, implying that each term in which it appears can be minimized for $x \rightarrow \infty$.

9.2.5 An example

Consider the example

$$\begin{array}{ll} \text{minimize} & x^{-1}y \\ \text{subject to} & x^2y^{-\frac{1}{2}} + 3y^{\frac{1}{2}}z^{-1} \leq 1, \\ & xy^{-1} = z^2, \\ & -x \leq -\frac{1}{10}, \\ & x \leq 3, \\ & x, y, z > 0, \end{array}$$

which is not a geometric optimization problem. However, using the obvious transformations we obtain the problem

$$\begin{array}{ll} \text{minimize} & x^{-1}y \\ \text{subject to} & x^2y^{-\frac{1}{2}} + 3y^{\frac{1}{2}}z^{-1} \leq 1, \\ & xy^{-1}z^{-2} \leq 1, \\ & x^{-1}yz^2 \leq 1, \\ & \frac{1}{10}x^{-1} \leq 1, \\ & \frac{1}{3}x \leq 1, \\ & x, y, z > 0, \end{array} \tag{9.31}$$

which is a geometric optimization problem.

9.2.6 Solving the example

The problem (9.31) can be defined and solved in the MOSEK toolbox as shown below.

```
% go2.m

c      = [1 1 3 1 1 0.1 1/3]';
```

```

a      = sparse([[ -1  1  0];
                 [ 2 -0.5 0];
                 [ 0 0.5 -1];
                 [ 1 -1 -2];
                 [-1 1 2];
                 [-1 0 0];
                 [ 1 0 0]]);

map     = [0 1 1 2 3 4 5]';
[res] = mskgpopt(c,a,map);

fprintf('\nPrimal optimal solution to original gp:');
fprintf(' %e',exp(res.sol.itr.xx));
fprintf('\n\n');

% Compute the optimal objective value and
% the constraint activities.
v = c.*exp(a*res.sol.itr.xx);

% Add appropriate terms together.
f = sparse(map+1,1:7,ones(size(map)))*v;

% First objective value. Then constraint values.
fprintf('Objective value: %e\n',log(f(1)));
fprintf('Constraint values:');
fprintf(' %e',log(f(2:end)));
fprintf('\n\n');

% Dual multipliers (should be negative)
fprintf('Dual variables (should be negative):');
fprintf(' %e',res.sol.itr.y);
fprintf('\n\n');

```

9.2.7 Exporting to a file

It's possible to write a geometric optimization problem to a file with the command:

```
mskgpwri(c,a,map,filename)
```

This file format is compatible with the `mskexpopt` command line tool. See the MOSEK Tools User's manual for details on `mskexpopt`. This file format can be useful for sending

debug information to MOSEK or for testing. It's also possible to read the above format with the command:

```
[c,a,map] = mskgpread(filename)
```

9.2.8 Further information

More information about geometric optimization problems is located in [\[11, 12, 15\]](#).

Chapter 10

Modelling

In this chapter we will discuss the following issues:

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

10.1 Linear optimization

A linear optimization problem can be written as

$$\begin{array}{llllll} \text{minimize} & & c^T x + c^f & & & \\ \text{subject to} & l^c \leq & Ax & \leq & u^c, & \\ & l^x \leq & x & \leq & u^x, & \end{array} \quad (10.1)$$

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 (10.1). If (10.1) has at least one primal feasible solution, then (10.1) is said to be (primal) feasible.

In case (10.1) does not have a feasible solution, the problem is said to be *(primal) infeasible*.

10.1.1 Duality for linear optimization

Corresponding to the primal problem (10.1), there is a dual problem

$$\begin{aligned}
 & \text{maximize} && (l^c)^T s_l^c - (u^c)^T s_u^c \\
 & && + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f \\
 & \text{subject to} && A^T y + s_l^c - s_u^c = c, \\
 & && -y + s_l^x - s_u^x = 0, \\
 & && s_l^c, s_u^c, s_l^x, s_u^x \geq 0.
 \end{aligned} \tag{10.2}$$

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 (10.2). If (10.2) has at least one feasible solution, then (10.2) is *(dual) feasible*, otherwise the problem is *(dual) infeasible*.

We will denote a solution

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

so that x is a solution to the primal problem (10.1), and

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

is a solution to the corresponding dual problem (10.2). A solution which is both primal and dual feasible is denoted a *primal-dual feasible solution*.

10.1.1.1 A primal-dual feasible solution

Let

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

be a primal-dual feasible solution, and let

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

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

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

where the first relation can be obtained by multiplying the dual constraints (10.2) by x and x^c respectively, and the second relation comes from the fact that each term in each sum is nonnegative. It follows that the primal objective will always be greater than or equal to the dual objective.

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

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

Please note that the duality gap will always be nonnegative.

10.1.1.2 An optimal solution

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

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

are satisfied.

If (10.1) 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.

10.1.1.3 Primal infeasible problems

If the problem (10.1) is infeasible (has no feasible solution), MOSEK will report a certificate of primal infeasibility: The dual solution reported is a certificate of infeasibility, and the primal solution is undefined.

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

$$\begin{aligned} & \text{maximize} && (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x \\ & \text{subject to} && A^T y + s_l^x - s_u^x = 0, \\ & && -y + s_l^c - s_u^c = 0, \\ & && s_l^c, s_u^c, s_l^x, s_u^x \geq 0. \end{aligned} \tag{10.3}$$

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

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

to (10.3) so that

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

Such a solution implies that (10.3) is unbounded, and that its dual is infeasible.

We note that the dual of (10.3) is a problem which constraints are identical to the constraints of the original primal problem (10.1): If the dual of (10.3) is infeasible, so is the original primal problem.

10.1.1.4 Dual infeasible problems

If the problem (10.2) is infeasible (has no feasible solution), MOSEK will report a certificate of dual infeasibility: The primal solution reported is a certificate of infeasibility, and the dual solution is undefined.

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

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && Ax - x^c = 0, \\ & && \bar{l}^c \leq x^c \leq \bar{u}^c, \\ & && \bar{l}^x \leq x \leq \bar{u}^x \end{aligned} \tag{10.4}$$

where

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

and

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

so that the objective value $c^T x$ is negative. Such a solution implies that (10.4) is unbounded, and that the dual of (10.4) is infeasible.

We note that the dual of (10.4) is a problem which constraints are identical to the constraints of the original dual problem (10.2): If the dual of (10.4) is infeasible, so is the original dual problem.

10.1.2 Primal and dual infeasible case

In case that both the primal problem (10.1) and the dual problem (10.2) are infeasible, MOSEK will report only one of the two possible certificates — which one is not defined (MOSEK returns the first certificate found).

10.2 Quadratic and quadratically constrained optimization

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

$$\begin{aligned} & \text{minimize} && \frac{1}{2}x^T Q^o x + c^T x + c^f \\ & \text{subject to} && l_k^c \leq \frac{1}{2}x^T Q^k x + \sum_{j=0}^{n-1} a_{k,i} x_j \leq u_k^c, \quad k = 0, \dots, m-1, \\ & && l^x \leq x \leq u^x, \quad j = 0, \dots, n-1, \end{aligned} \quad (10.5)$$

where the convexity requirement implies that

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

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

10.2.1 A general recommendation

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

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

See Section 10.3.3.1 for further details.

10.2.2 Reformulating as a separable quadratic problem

The simplest quadratic optimization problem is

$$\begin{aligned} & \text{minimize} && 1/2x^T Qx + c^T x \\ & \text{subject to} && Ax = b, \\ & && x \geq 0. \end{aligned} \tag{10.6}$$

The problem (10.6) is said to be a separable problem if Q is a diagonal matrix or, in other words, if the quadratic terms in the objective all have this form

$$x_j^2$$

instead of this form

$$x_j x_i.$$

The separable form has the following advantages:

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

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

$$Q = F^T F. \tag{10.7}$$

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

Using (10.7), the problem (10.6) can be reformulated as

$$\begin{aligned} & \text{minimize} && 1/2y^T Iy + c^T x \\ & \text{subject to} && Ax = b, \\ & && Fx - y = 0, \\ & && x \geq 0. \end{aligned} \tag{10.8}$$

The problem (10.8) is also a quadratic optimization problem and has more constraints and variables than (10.6). However, the problem is separable. Normally, if F has fewer rows than columns, it is worthwhile to reformulate as a separable problem. Indeed consider the extreme case where F has one dense row and hence Q will be a dense matrix.

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

$$1/2x^T (F^T F)x \leq b \tag{10.9}$$

where F is a matrix and b is a scalar. (10.9) can be reformulated as

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

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

Next, consider the constraint

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

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

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

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

10.3 Conic optimization

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

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

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

$$\begin{aligned} &\text{minimize} && c^T x + c^f \\ &\text{subject to} && \begin{array}{lll} l^c & \leq & Ax & \leq & u^c, \\ l^x & \leq & x & \leq & u^x, \end{array} \\ &&& x \in \mathcal{C}. \end{aligned} \tag{10.10}$$

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

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

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

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

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

MOSEK supports only a limited number of cones, specifically

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

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

- \mathbb{R} set:

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

- Quadratic cone:

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

- Rotated quadratic cone:

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

Although these cones may seem to provide only limited expressive power they can be used to model a large range of problems as demonstrated in Section 10.3.3.

10.3.1 Duality for conic optimization

The dual problem corresponding to the conic optimization problem (10.10) is given by

$$\begin{aligned} & \text{maximize} && (l^c)^T s_l^c - (u^c)^T s_u^c \\ & && + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f \\ & \text{subject to} && A^T y + s_l^x - s_u^x + s_n^x = c, \\ & && -y + s_l^c - s_u^c = 0, \\ & && s_l^c, s_u^c, s_l^x, s_u^x \geq 0, \\ & && s_n^x \in \mathcal{C}^* \end{aligned} \tag{10.11}$$

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

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

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:

- \mathbb{R} set:

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

- Quadratic cone:

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

- Rotated quadratic cone:

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

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

10.3.2 Infeasibility

In case MOSEK finds a problem to be infeasible it reports a certificate of the infeasibility. This works exactly as for linear problems (see Sections 10.1.1.3 and 10.1.1.4).

10.3.3 Examples

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

10.3.3.1 Quadratic objective and constraints

From Section 10.2.2 we know that any convex quadratic problem can be stated on the form

$$\begin{aligned} & \text{minimize} && 0.5 \|Fx\|^2 + c^T x, \\ & \text{subject to} && 0.5 \|Gx\|^2 + a^T x \leq b, \end{aligned} \tag{10.12}$$

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

Problem (10.12) can be reformulated as

$$\begin{aligned} & \text{minimize} && 0.5 \|t\|^2 + c^T x, \\ & \text{subject to} && 0.5 \|z\|^2 + a^T x \leq b, \\ & && Fx - t = 0, \\ & && Gx - z = 0 \end{aligned} \tag{10.13}$$

after the introduction of the new variables t and z . It is easy to convert this problem to a

conic quadratic optimization problem, i.e.

$$\begin{aligned}
& \text{minimize} && v + c^T x, \\
& \text{subject to} && p + a^T x = b, \\
& && Fx - t = 0, \\
& && Gx - z = 0, \\
& && w = 1, \\
& && q = 1, \\
& && \|t\|^2 \leq 2vw, \quad v, w \geq 0, \\
& && \|z\|^2 \leq 2pq, \quad p, q \geq 0.
\end{aligned} \tag{10.14}$$

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

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

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

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

$$\begin{aligned}
& \text{minimize} && v + c^T F^{-1}t, \\
& \text{subject to} && p + a^T F^{-1}t = b, \\
& && GF^{-1}t - z = 0, \\
& && w = 1, \\
& && q = 1, \\
& && \|t\|^2 \leq 2vw, \quad v, w \geq 0, \\
& && \|z\|^2 \leq 2pq, \quad p, q \geq 0.
\end{aligned} \tag{10.15}$$

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

10.3.3.2 Minimizing a sum of norms

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

$$\begin{aligned}
& \text{minimize} && \sum_{i=1}^k \|x^i\| \\
& \text{subject to} && Ax = b,
\end{aligned} \tag{10.16}$$

where

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

This problem is equivalent to

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^k z_i \\ & \text{subject to} && Ax = b, \\ & && \|x^i\| \leq z_i, \quad i = 1, \dots, k, \end{aligned} \tag{10.17}$$

which in turn is equivalent to

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^k z_i \\ & \text{subject to} && Ax = b, \\ & && (z_i, x^i) \in \mathcal{C}_i, \quad i = 1, \dots, k \end{aligned} \tag{10.18}$$

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

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

The dual problem corresponding to (10.18) is

$$\begin{aligned} & \text{maximize} && b^T y \\ & \text{subject to} && A^T y + s = c, \\ & && t_i = 1, \quad i = 1, \dots, k, \\ & && (t_i, s^i) \in \mathcal{C}_i, \quad i = 1, \dots, k \end{aligned} \tag{10.19}$$

where

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

This problem is equivalent to

$$\begin{aligned} & \text{maximize} && b^T y \\ & \text{subject to} && A^T y + s = c, \\ & && \|s^i\|_2^2 \leq 1, \quad i = 1, \dots, k. \end{aligned} \tag{10.20}$$

Please note that in this case the dual problem can be reduced to an “ordinary” convex quadratically constrained optimization problem due to the special structure of the primal problem. In some cases it turns out that it is much better to solve the dual problem (10.19) rather than the primal problem (10.18).

10.3.3.3 Modelling polynomial terms using conic optimization

Generally an arbitrary polynomial term of the form

$$fx^g$$

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

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

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

Now, a constraint of the form

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

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

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

which in turn is equivalent to

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

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

E.g., consider the problem

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

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

$$\begin{aligned} &\text{minimize} && c^T x \\ &\text{subject to} && \sum_{j=1}^n f_j z_j = b, \\ &&& v_j = \sqrt{2}, \quad j = 1, \dots, n, \\ &&& v_j^2 \leq 2z_j x_j, \quad j = 1, \dots, n, \\ &&& x, z \geq 0, \end{aligned} \tag{10.21}$$

because

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

implies that

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

The problem (10.21) is a conic quadratic optimization problem having n 3-dimensional rotated quadratic cones.

The next example is the constraint

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

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

$$\begin{aligned}t^2 &\leq 2xz, \\ z &= 0.5, \\ x, z, &\geq 0.\end{aligned}\tag{10.22}$$

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

$$\begin{aligned}x^{1.5} &\leq t, \\ 0 &\leq x.\end{aligned}\tag{10.23}$$

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

$$\begin{aligned}x^2 &\leq 2st, \\ s, t &\geq 0.\end{aligned}\tag{10.24}$$

Now, if we can make sure that

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

then we have the desired result since this implies that

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

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

Let

$$\begin{aligned}x^2 &\leq 2st, \\ w^2 &\leq 2vr, \\ x &= v, \\ s &= w, \\ r &= \frac{1}{8}, \\ s, t, v, r &\geq 0,\end{aligned}\tag{10.25}$$

then

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

Moreover,

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

leading to the conclusion that

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

(10.25) is a conic reformulation which is equivalent to (10.23). Please note that the $x \geq 0$ constraint does not appear explicitly in (10.24) and (10.25), but implicitly since $x = v \geq 0$.

As we shall see next, any polynomial term of the form x^g where g is a positive rational number can be represented using conic quadratic constraints [2, pp. 12-13], [14].

10.3.3.4 Optimization with rational polynomials

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

Following Ben-Tal et al. [14, p. 105] we use an intermediate result, namely that the set

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

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

$$s \leq (2^{12} \cdot y_1 \cdot y_2 \cdot y_3 \cdot y_4 \cdot y_5 \cdot y_6 \cdot y_7 \cdot y_8)^{1/8} \quad (10.26)$$

as

$$s^8 \leq (2^{12} \cdot y_1 \cdot y_2 \cdot y_3 \cdot y_4 \cdot y_5 \cdot y_6 \cdot y_7 \cdot y_8) \quad (10.27)$$

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

$$y_{11}^2 \leq 2y_1 y_2, \quad y_{12}^2 \leq 2y_3 y_4, \quad y_{13}^2 \leq 2y_5 y_6, \quad y_{14}^2 \leq 2y_7 y_8, \quad (10.28)$$

$$y_{21}^2 \leq 2y_{11} y_{12}, \quad y_{22}^2 \leq 2y_{13} y_{14}, \quad (10.29)$$

and finally

$$s^2 \leq 2y_{21} y_{22}. \quad (10.30)$$

By simple substitution we see that (10.30) and (10.27) are equivalent, and since (10.30) involves only a set of simple rotated conic constraints then the original constraint (10.26) can be represented using only rotated conic constraints.

10.3.3.5 Convex increasing power functions

Using the intermediate result in section 10.3.3.4 we can include convex power functions with positive rational powers, i.e., constraints of the form

$$x^{p/q} \leq t, \quad x \geq 0$$

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

$$x^{5/3} \leq t, \quad x \geq 0.$$

We rewrite it as

$$x^8 \leq x^3 t^3, \quad x \geq 0$$

which in turn is equivalent to

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

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

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

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

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

$$x^{2^l} \leq x^{2^l - p} t^q, \quad x \geq 0 \quad \Longleftrightarrow \quad x^{p/q} \leq t, \quad x \geq 0.$$

10.3.3.6 Decreasing power functions

We can also include decreasing power functions with positive rational powers

$$x^{-p/q} \leq t, \quad x \geq 0$$

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

$$x^{-5/2} \leq t, \quad x \geq 0,$$

or equivalently

$$1 \leq x^5 t^2, \quad x \geq 0,$$

which, in turn, can be rewritten as

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

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

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

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

$$1 \leq x^p t^q, \quad x \geq 0 \quad \Longleftrightarrow \quad x^{-p/q} \leq t, \quad x \geq 0.$$

10.3.3.7 Minimizing general polynomials

Using the formulations in section 10.3.3.5 and section 10.3.3.6 it is straightforward to minimize general polynomials. For example, we can minimize

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

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

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

which is equivalent to the quadratic conic optimization problem

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

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

10.3.3.8 Further reading

If you want to learn more about what can be modeled as a conic optimization problem we recommend the references [2, 14, 19].

10.3.4 Potential pitfalls in conic optimization

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

10.3.4.1 Non-attainment in the primal problem

Consider the example

$$\begin{aligned}
 &\text{minimize} && z \\
 &\text{subject to} && 2yz \geq x^2, \\
 & && x = \sqrt{2}, \\
 & && y, z \geq 0,
 \end{aligned} \tag{10.31}$$

which corresponds to the problem

$$\begin{aligned}
 &\text{minimize} && \frac{1}{y} \\
 &\text{subject to} && y \geq 0.
 \end{aligned} \tag{10.32}$$

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

10.3.4.2 Non-attainment in the dual problem

Next, consider the example

$$\begin{aligned}
 &\text{minimize} && x_4 \\
 &\text{subject to} && x_3 + x_4 = 1, \\
 & && x_1 = 0, \\
 & && x_2 = 1, \\
 & && 2x_1x_2 \geq x_3^2, \\
 & && x_1, x_2 \geq 0,
 \end{aligned} \tag{10.33}$$

which has the optimal solution

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

implying that the optimal primal objective value is 1.

Now, the dual problem corresponding to (10.33) is

$$\begin{aligned}
 &\text{maximize} && y_1 + y_3 \\
 &\text{subject to} && y_2 + s_1 = 0, \\
 & && y_3 + s_2 = 0, \\
 & && y_1 + s_3 = 0, \\
 & && y_1 = 1, \\
 & && 2s_1s_2 \geq s_3^2, \\
 & && s_1, s_2 \geq 0.
 \end{aligned} \tag{10.34}$$

Therefore,

$$y_1^* = 1$$

and

$$s_3^* = -1.$$

This implies that

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

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

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

implying that the optimal dual objective value is 1, however, this is never attained. Hence, no primal-dual bounded optimal solution with zero duality gap exists. Of course it is possible to find a primal-dual feasible solution such that the duality gap is close to zero, but then s_1^* will be similarly large. This is likely to make the problem (10.33) hard to solve.

An inspection of the problem (10.33) reveals the constraint $x_1 = 0$, which implies that $x_3 = 0$. If we either add the redundant constraint

$$x_3 = 0$$

to the problem (10.33) or eliminate x_1 and x_3 from the problem it becomes easy to solve.

10.4 Nonlinear convex optimization

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

$$\begin{aligned} &\text{minimize} && f(x) + c^T x \\ &\text{subject to} && g(x) + Ax - x^c = 0, \\ & && l^c \leq x^c \leq u^c, \\ & && l^x \leq x \leq u^x, \end{aligned} \tag{10.35}$$

where

- m is the number of constraints.
- n is the number of decision variables.
- $x \in \mathbb{R}^n$ is a vector of decision variables.
- $x^c \in \mathbb{R}^m$ is a vector of constraints or slack variables.
- $c \in \mathbb{R}^n$ is the linear part objective function.
- $A \in \mathbb{R}^{m \times n}$ is the constraint matrix.

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

This means that the i th constraint has the form

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

when the x_i^c variable has been eliminated.

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

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

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

10.4.1 Duality

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

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

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

and the dual problem is given by

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

which is equivalent to

$$\begin{aligned}
& \text{maximize} && f(x) - y^T g(x) - x^T (\nabla f(x)^T - \nabla g(x)^T y) \\
& && + ((l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f \\
& \text{subject to} && -\nabla f(x)^T + A^T y + \nabla g(x)^T y + s_l^x - s_u^x = c, \\
& && -y + s_l^c - s_u^c = 0, \\
& && s_l^c, s_u^c, s_l^x, s_u^x \geq 0.
\end{aligned} \tag{10.36}$$

10.5 Recommendations

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

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

1. Sparsity is very important. The constraint matrix A is assumed to be a sparse matrix, where sparse means that it contains many zeros (typically less than 10% non-zeros). Normally, when A is sparser, less memory is required to store the problem and it can be solved faster.
2. Avoid large bounds as these can introduce all sorts of numerical problems. Assume that a variable x_j has the bounds

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

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

3. Avoid large penalty terms in the objective, i.e. do not have large terms in the linear part of the objective function. They will most likely cause numerical problems.
4. On a computer all computations are performed in finite precision, which implies that

$$1 = 1 + \varepsilon$$

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

5. Decreasing the number of variables or constraints does not *necessarily* make it easier to solve a problem. In certain cases, i.e. in nonlinear optimization, it may be a good idea to introduce more constraints and variables if it makes the model separable. Furthermore, a big but sparse problem may be advantageous compared to a smaller but denser problem.
6. Try to avoid linearly dependent rows among the linear constraints. Network flow problems and multi-commodity network flow problems, for example, often contain one or more linearly dependent rows.
7. Finally, it is recommended to consult some of the papers about preprocessing to get some ideas about efficient formulations. See e.g. [3, 5, 17, 18].

10.5.1 Avoid near infeasible models

Consider the linear optimization problem

$$\begin{aligned}
 & \text{minimize} \\
 & \text{subject to} \quad \begin{aligned} x + y &\leq 10^{-10} + \alpha, \\ 1.0e4x + 2.0e4y &\geq 10^{-6}, \\ x, y &\geq 0. \end{aligned}
 \end{aligned} \tag{10.37}$$

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

10.6 Examples continued

10.6.1 The absolute value

Assume that we have a constraint for the form

$$|f^T x + g| \leq b \tag{10.38}$$

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

It is easy to verify that the constraint (10.38) is equivalent to

$$-b \leq f^T x + g \leq b \tag{10.39}$$

which is a set of ordinary linear inequality constraints.

Please note that equalities involving an absolute value such as

$$|x| = 1$$

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

10.6.2 The Markowitz portfolio model

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

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

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

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

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

The model deals with two central quantities:

- Expected return:

$$E[r^T(w^0 + x)] = \bar{r}^T(w^0 + x).$$

- Variance (Risk):

$$V[r^T(w^0 + x)] = (w^0 + x)^T \Sigma (w^0 + x).$$

By definition Σ is positive semi-definite and

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

where L is **any** matrix such that

$$\Sigma = LL^T$$

A low rank of Σ is advantageous from a computational point of view. A valid L can always be computed as the Cholesky factorization of Σ .

10.6.2.1 Minimizing variance for a given return

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

$$\begin{aligned} & \text{minimize} && V[r^T(w^0 + x)] \\ & \text{subject to} && E[r^T(w^0 + x)] = t, \\ & && e^T x = 0, \end{aligned} \tag{10.40}$$

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

$$\begin{aligned} & \text{minimize} && (w^0 + x)^T \Sigma (w^0 + x) \\ & \text{subject to} && \bar{r}^T (w^0 + x) = t, \\ & && e^T x = 0. \end{aligned} \tag{10.41}$$

10.6.2.2 Conic quadratic reformulation

An equivalent conic quadratic reformulation is given by:

$$\begin{aligned} & \text{minimize} && f \\ & \text{subject to} && \Sigma^{\frac{1}{2}}(w^0 + x) - g = 0, \\ & && \bar{r}^T (w^0 + x) = t, \\ & && e^T x = 0, \\ & && f \geq \|g\|. \end{aligned} \tag{10.42}$$

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

10.6.2.3 Transaction costs with market impact term

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

$$\text{commission} + \frac{\text{bid}}{\text{ask}} - \text{spread} + \theta \sqrt{\frac{\text{trade volume}}{\text{daily volume}}}, \tag{10.43}$$

and that it is important to incorporate these into the model.

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

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

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

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

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

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

This leads us to the model:

$$\begin{aligned} & \text{minimize} && f \\ & \text{subject to} && \Sigma^{\frac{1}{2}}(w^0 + x) - g = 0, \\ & && \bar{r}^T(w^0 + x) = t, \\ & && e^T x + e^T y = 0, \\ & && |x_j|(m_j|x_j|^{\frac{1}{2}}) \leq y_j, \\ & && f \geq \|g\|. \end{aligned} \tag{10.44}$$

Now, defining the variable transformation

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

we obtain

$$\begin{aligned} & \text{minimize} && f \\ & \text{subject to} && \Sigma^{\frac{1}{2}}(w^0 + x) - g = 0, \\ & && \bar{r}^T(w^0 + x) = t, \\ & && e^T x + m^T \bar{y} = 0, \\ & && |x_j|^{3/2} \leq \bar{y}_j, \\ & && f \geq \|g\|. \end{aligned} \tag{10.45}$$

As shown in Section 10.3.3.3 the set

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

can be modeled by

$$\begin{aligned} x_j & \leq z_j, \\ -x_j & \leq z_j, \\ z_j^2 & \leq 2s_j \bar{y}_j, \\ u_j^2 & \leq 2v_j q_j, \\ z_j & = v_j, \\ s_j & = u_j, \\ q_j & = \frac{1}{8}, \\ q_j, s_j, \bar{y}_j, v_j, q_j & \geq 0. \end{aligned} \tag{10.46}$$

10.6.2.4 Further reading

For further reading please see [20] in particular, and [23] and [1], which also contain relevant material.

Chapter 11

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.

11.1 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.

11.1.1 Presolve

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

- remove redundant constraints,

- eliminate fixed variables,
- remove linear dependencies,
- substitute out free variables, and
- reduce the size of the optimization problem in general.

After the presolved problem has been optimized the solution is automatically postsolved so that the returned solution is valid for the original problem. Hence, the presolve is completely transparent. For further details about the presolve phase, please see [3, 5].

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.

11.1.1.1 Eliminator

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

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

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`.

11.1.1.2 Linear dependency checker

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

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

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`.

11.1.2 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.

11.1.3 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.

11.1.4 Using multiple CPU's

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

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

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

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

11.2 Linear optimization

11.2.1 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`.

11.2.2 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

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && Ax = b, \\ & && x \geq 0. \end{aligned} \tag{11.1}$$

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 (11.1) 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

$$\begin{aligned} Ax - b\tau &= 0, \\ A^T y + s - c\tau &= 0, \\ -c^T x + b^T y - \kappa &= 0, \\ x, s, \tau, \kappa &\geq 0, \end{aligned} \tag{11.2}$$

where y and s correspond to the dual variables in (11.1), and τ and κ are two additional scalar variables. Note that the homogeneous model (11.2) 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 (11.2) 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

$$\begin{aligned} A \frac{x^*}{\tau^*} &= b, \\ A^T \frac{y^*}{\tau^*} + \frac{s^*}{\tau^*} &= c, \\ -c^T \frac{x^*}{\tau^*} + b^T \frac{y^*}{\tau^*} &= 0, \\ x^*, s^*, \tau^*, \kappa^* &\geq 0. \end{aligned} \tag{11.3}$$

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

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

is a primal-dual optimal solution.

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

$$\begin{aligned} Ax^* &= 0, \\ A^T y^* + s^* &= 0, \\ -c^T x^* + b^T y^* &= \kappa^*, \\ x^*, s^*, \tau^*, \kappa^* &\geq 0. \end{aligned} \tag{11.4}$$

This implies that at least one of

$$-c^T x^* > 0 \tag{11.5}$$

or

$$b^T y^* > 0 \tag{11.6}$$

is satisfied. If (11.5) is satisfied then x^* is a certificate of dual infeasibility, whereas if (11.6) is satisfied then y^* is a certificate of dual infeasibility.

In summary, by computing an appropriate solution to the homogeneous model, all information required for a solution to the original problem is obtained. A solution to the homogeneous model can be computed using a primal-dual interior-point algorithm [10].

11.2.2.1 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

$$\begin{aligned} \left\| A \frac{x^k}{\tau^k} - b \right\| &\leq \varepsilon_p (1 + \|b\|), \\ \left\| A^T \frac{y^k}{\tau^k} + \frac{s^k}{\tau^k} - c \right\| &\leq \varepsilon_d (1 + \|c\|), \text{ and} \\ \min \left(\frac{(x^k)^T s^k + \tau^k \kappa^k}{(\tau^k)^2}, \left| \frac{c^T x^k}{\tau^k} - \frac{b^T y^k}{\tau^k} \right| \right) &\leq \varepsilon_g \max \left(1, \left| \frac{c^T x^k}{\tau^k} \right| \right), \end{aligned} \quad (11.7)$$

the interior-point optimizer is terminated and

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

is reported as the primal-dual optimal solution. The interpretation of (11.7) 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

$$-\varepsilon_i c^T x^k > \frac{\|c\|}{\max(\|b\|, 1)} \|Ax^k\| \quad (11.8)$$

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

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

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 11.1: Parameters employed in termination criterion.

and define

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

It is easy to verify that

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

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

Finally, if

$$\varepsilon_i b^T y^k \geq \frac{\|b\|}{\max(1, \|c\|)} \|A^T y^k + s^k\| \quad (11.9)$$

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

It is possible to adjust the tolerances ε_p , ε_d , ε_g and ε_i using parameters; see table 11.1 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 (11.7) reveals that quality of the solution is dependent on $\|b\|$ and $\|c\|$; the smaller the norms are, the better the solution accuracy.

The interior-point method as implemented by MOSEK will converge toward optimality and primal and dual feasibility at the same rate [10]. This means that if the optimizer is stopped prematurely then it is very unlikely that either the primal or dual solution is feasible. Another consequence is that in most cases all the tolerances, ε_p , ε_d and ε_g , has to be relaxed together to achieve an effect.

The basis identification discussed in section 11.2.2.2 requires an optimal solution to work well; hence basis identification should be turned off if the termination criterion is relaxed.

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

11.2.2.2 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 [7].

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

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

- `MSK_IPAR_INTPNT_BASIS`,
- `MSK_IPAR_BI_IGNORE_MAX_ITER`, and
- `MSK_IPAR_BI_IGNORE_NUM_ERROR`

controls when basis identification is performed.

11.2.2.3 The interior-point log

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

```

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

```

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

Using the same notation as in section 11.2.2 the columns of the iteration log has the following meaning:

- ITE: Iteration index.
- PFEAS: $\|Ax^k - b\tau^k\|$. The numbers in this column should converge monotonically towards to zero.

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

11.2.3 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 11.2.4 for a discussion.

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

11.2.3.1 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 (10.1) and (10.2) 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`.

11.2.3.2 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.

11.2.3.3 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.

11.2.4 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.

11.2.5 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.

11.3 Linear network optimization

11.3.1 Network flow problems

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

To use the network simplex optimizer, do the following:

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

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

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

Table 11.2: Parameters employed in termination criterion.

11.3.2 Embedded network problems

Often problems contains both large parts with network structure and some non-network constraints or variables — such problems are said to have *embedded network structure*.

If the procedure described in section 11.3.1 is applied, MOSEK will attempt to exploit this structure to speed up the optimization.

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

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

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

11.4 Conic optimization

11.4.1 The interior-point optimizer

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

11.4.1.1 Interior-point termination criteria

The parameters controlling when the conic interior-point optimizer terminates are shown in Table 11.2.

11.5 Nonlinear convex optimization

11.5.1 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 [8, 9].

11.5.1.1 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.

11.5.1.2 The differentiability 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.

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.

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

Table 11.3: Parameters employed in termination criteria.

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

$$f(x) = \frac{1}{x} + \frac{1}{1-x}. \quad (11.10)$$

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, (11.10) can be reformulated as follows

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

11.5.1.3 Interior-point termination criteria

The parameters controlling when the general convex interior-point optimizer terminates are shown in Table 11.3.

11.6 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.

11.6.1 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.

11.6.2 The parallelized interior-point optimizer

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

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

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

11.6.3 The concurrent optimizer

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

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

By setting the

`MSK_IPAR_OPTIMIZER`

parameter to

`MSK_OPTIMIZER_CONCURRENT`

the concurrent optimizer chosen.

The number of optimizers used in parallel is determined by the

`MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS`.

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

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

Table 11.4: Default priorities for optimizer selection in concurrent optimization.

is shown in Table 11.6.3. For example, setting the `MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS` parameter to 2 tells the concurrent optimizer to apply the two optimizers with highest priorities: In the default case that means the interior-point optimizer and one of the simplex optimizers.

11.7 Understanding solution quality

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

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

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

11.7.1 The solution summary

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

Normally, in the optimization toolbox for MATLAB the solution summary is displayed immediately after the optimization has completed.

11.7.1.1 The optimal case

The solution summary has the format

```

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

```

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

Assumeing that we are solving a linear optimization problem and referring to the problems (10.1) and (10.2), the interpretation of the solution summary is as follows:

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

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

11.7.1.2 The primal infeasible case

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

```

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

```

It is known that if the problem is primal infeasible then an infeasibility certificate exists, which is a solution to the problem (10.3) having a positive objective value. Note that the primal solution plays no role and only the dual solution is used to specify the certificate.

Therefore, in the primal infeasible case the solution summary should report how good the dual solution is to the problem (10.3). The interpretation of the solution summary is as follows:

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

Please note that

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

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

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

Please see Section 13.2 for more information about certificates of infeasibility.

Chapter 12

The optimizer for mixed integer problems

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

- linear,
- quadratic and
- conic

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

Readers unfamiliar with integer optimization are strongly recommended to consult some relevant literature, e.g. the book [26] by Wolsey is a good introduction to integer optimization.

12.1 Some notation

In general, an integer optimization problem has the form

$$\begin{aligned} z^* = & \text{minimize} && c^T x \\ & \text{subject to} && l^c \leq Ax \leq u^c, \\ & && l^x \leq x \leq u^x, \\ & && x_j \in \mathcal{Z}, \quad \forall j \in \mathcal{J}, \end{aligned} \tag{12.1}$$

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

$$\begin{aligned} \underline{z} = & \text{minimize} && c^T x \\ & \text{subject to} && l^c \leq Ax \leq u^c, \\ & && l^x \leq x \leq u^x \end{aligned} \tag{12.2}$$

i.e. we ignore the constraint

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

Moreover, let \hat{x} be any feasible solution to (12.1) and define

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

It should be obvious that

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

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

12.2 An important fact about integer optimization problems

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

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

12.3 How the integer optimizer works

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

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

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

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

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

12.3.1 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.

12.3.2 Heuristic

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

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

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

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

12.3.3 The optimization phase

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

12.4 Termination criterion

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

A candidate solution, i.e. a solution to (12.2), is said to be an integer feasible solution if the criterion

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

is satisfied. Hence, such a solution is defined as a feasible solution to (12.1).

Tolerance	Parameter name
δ_1	<code>MSK_DPAR_MIO_TOL_ABS_RELAX_INT</code>
δ_2	<code>MSK_DPAR_MIO_TOL_REL_RELAX_INT</code>
δ_3	<code>MSK_DPAR_MIO_TOL_ABS_GAP</code>
δ_4	<code>MSK_DPAR_MIO_TOL_REL_GAP</code>
δ_5	<code>MSK_DPAR_MIO_NEAR_TOL_ABS_GAP</code>
δ_6	<code>MSK_DPAR_MIO_NEAR_TOL_REL_GAP</code>

Table 12.1: Integer optimizer tolerances.

Parameter name	Delayed	Explanation
<code>MSK_IPAR_MIO_MAX_NUM_BRANCHES</code>	Yes	Maximum number of branches allowed.
<code>MSK_IPAR_MIO_MAX_NUM_RELAXS</code>	Yes	Maximum number of relaxations allowed.
<code>MSK_IPAR_MIO_MAX_NUM_SOLUTIONS</code>	Yes	Maximum number of feasible integer solutions allowed.

Table 12.2: Parameters affecting the termination of the integer optimizer.

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.

The δ tolerances can be specified using parameters — see Table 12.1. If an optimal solution cannot be located within a reasonable time, it may be advantageous to employ a relaxed termination criterion after some time. Whenever the integer optimizer locates an integer feasible solution and has spent at 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

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

is satisfied. If it is satisfied, the optimizer will report that the candidate solution is **near optimal** and then terminate. All δ tolerances can be adjusted using suitable parameters — see Table 12.1. In Table 12.2 some other parameters affecting the integer optimizer termination criterion are shown. Please note that if the effect of a parameter is delayed, the associated termination criterion is applied only after some time, specified by the `MSK_DPAR_MIO_DISABLE_TERM_TIME` parameter.

12.5 How to speed up the solution process

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

- Relax the termination criterion: In case the run time is not acceptable, the first thing to do is to relax the termination criterion — see Section 12.4 for details.
- Specify a good initial solution: In many cases a good feasible solution is either known or easily computed using problem specific knowledge. If a good feasible solution is known, it is usually worthwhile to use this as a starting point for the integer optimizer.
- Improve the formulation: A mixed-integer optimization problem may be impossible to solve in one form and quite easy in another form. However, it is beyond the scope of this manual to discuss good formulations for mixed-integer problems. For discussions on this topic see for example [26].

12.6 Understanding solution quality

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

- The solution status key returned by MOSEK.
- The *optimality gap*: A 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 guaranteed 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`.

12.6.1 Solutionsummary

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

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

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

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

Chapter 13

The analyzers

13.1 The problem analyzer

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

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

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

The problem analyzer is run using the `mosekopt('anapro')` command and produces something similar to the following (this is the problem analyzer's survey of the `aflow30a` problem from the MIPLIB 2003 collection, see Appendix G 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.

13.1.1 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 G; if all entities in a section turn out to be of the same kind, the number will be replaced by **all** for brevity.

13.1.2 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 coefficients

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.

13.1.3 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 G.

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

13.1.4 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).

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

13.1.6 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.

13.2 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.

13.2.1 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 13.1.

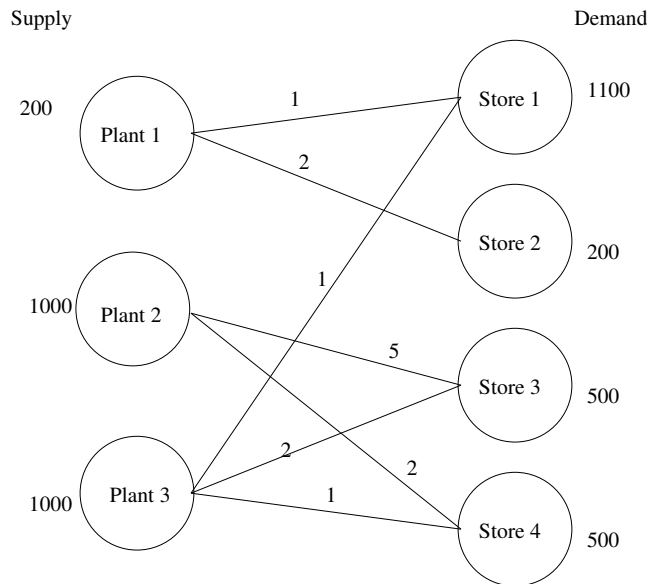


Figure 13.1: Supply, demand and cost of transportation.

The problem represented in figure 13.1 is infeasible, since the total demand

$$2300 = 1100 + 200 + 500 + 500 \quad (13.1)$$

exceeds the total supply

$$2200 = 200 + 1000 + 1000 \quad (13.2)$$

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

be formulated as the LP:

$$\begin{array}{llllllllllll}
 \text{minimize} & x_{11} & + & 2x_{12} & + & 5x_{23} & + & 2x_{24} & + & x_{31} & + & 2x_{33} & + & x_{34} \\
 \text{subject to} & x_{11} & + & x_{12} & & & & & & & & & & \leq 200, \\
 & & & & & x_{23} & + & x_{24} & & & & & & \leq 1000, \\
 & & & & & & & & x_{31} & + & x_{33} & + & x_{34} & \leq 1000, \\
 & x_{11} & & & & & & & + & x_{31} & & & & = 1100, \\
 & & x_{12} & & & & & & & & & & & = 200, \\
 & & & x_{23} & + & & & & & & x_{33} & & & = 500, \\
 & & & & & x_{24} & + & & & & & x_{34} & = 500, \\
 & x_{ij} & \geq 0.
 \end{array} \tag{13.3}$$

Solving the problem (13.3) using MOSEK will result in a solution, a solution status and a problem status. Among the log output from the execution of MOSEK on the above problem are the lines:

```

Basic solution
Problem status : PRIMAL_INFEASIBLE
Solution status : PRIMAL_INFEASIBLE_CER

```

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

13.2.2 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 13.2.4) may assist you in finding the constraints causing the infeasibility.

Possible ways of relaxing your problem include:

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

- Removing suspected constraints from the problem.

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

$$x_{12} = 200 \tag{13.4}$$

makes the problem feasible.

13.2.3 Locating the cause of dual infeasibility

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

$$\begin{array}{ll} \text{minimize} & x_1 \\ \text{subject to} & x_1 \leq 5. \end{array} \tag{13.5}$$

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

- Adding upper or lower bounds on variables or constraints.
- Removing variables.
- Changing the objective.

13.2.3.1 A cautious note

The problem

$$\begin{array}{ll} \text{minimize} & 0 \\ \text{subject to} & 0 \leq x_1, \\ & x_j \leq x_{j+1}, \quad j = 1, \dots, n-1, \\ & x_n \leq -1 \end{array} \tag{13.6}$$

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

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

13.2.4 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.

13.2.4.1 Example: Primal infeasibility

We will reuse the example (13.3) located in `infeas.lp`:

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

Using the command line

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

MOSEK produces the following infeasibility report

MOSEK PRIMAL INFEASIBILITY REPORT.

Problem status: The problem is primal infeasible

The following constraints are involved in the primal infeasibility.

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

The following bound constraints are involved in the infeasibility.

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

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

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

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

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

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

which is an optimization problem. This problem is identical to (13.3), except that the objective and some of the constraints and bounds have been removed. Executing the command

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

demonstrates that the reduced problem is **primal infeasible**. Since the reduced problem is usually smaller than original problem, it should be easier to locate the cause of the infeasibility in this rather than in the original (13.3).

13.2.4.2 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

$$c^T x^* > 0. \quad (13.7)$$

For a minimization problem this inequality would have been reversed — see (13.19).

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

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

- Put a lower bound in **y3**. This will directly invalidate the certificate of dual infeasibility.
- Increase the object coefficient of **y3**. Changing the coefficients sufficiently will invalidate the inequality (13.7) and thus the certificate.
- Put lower bounds on **x11** or **x31**. This will directly invalidate the certificate of infeasibility.

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

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

13.2.5 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

$$\begin{aligned} & \text{minimize} && c^T x + c^f \\ & \text{subject to} && \begin{array}{lll} l^c & \leq & Ax & \leq & u^c, \\ l^x & \leq & x & \leq & u^x \end{array} \end{aligned} \quad (13.8)$$

where the corresponding dual problem is

$$\begin{aligned} & \text{maximize} && (l^c)^T s_l^c - (u^c)^T s_u^c \\ & && + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f \\ & \text{subject to} && \begin{array}{lll} A^T y + s_l^x - s_u^x & = & c, \\ -y + s_l^c - s_u^c & = & 0, \\ s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0. \end{array} \end{aligned} \quad (13.9)$$

We use the convention 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

$$l_j^x = -\infty \Rightarrow (s_l^x)_j = 0 \quad (13.10)$$

13.2.6 The certificate of primal infeasibility

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

$$\begin{aligned} & \text{maximize} && (l^c)^T s_l^c - (u^c)^T s_u^c \\ & && + (l^x)^T s_l^x - (u^x)^T s_u^x \\ & \text{subject to} && A^T y + s_l^x - s_u^x = 0, \\ & && -y + s_l^c - s_u^c = 0, \\ & && s_l^c, s_u^c, s_l^x, s_u^x \geq 0. \end{aligned} \quad (13.11)$$

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

$$(l^c)^T s_l^{c*} - (u^c)^T s_u^{c*} + (l^x)^T s_l^{x*} - (u^x)^T s_u^{x*} > 0 \quad (13.12)$$

and

$$\begin{aligned} A^T y + s_l^{x*} - s_u^{x*} &= 0, \\ -y + s_l^{c*} - s_u^{c*} &= 0, \\ s_l^{c*}, s_u^{c*}, s_l^{x*}, s_u^{x*} &\geq 0. \end{aligned} \quad (13.13)$$

The well-known Farkas Lemma tells us that (13.8) is infeasible if and only if a certificate of primal infeasibility exists.

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

$$(s_l^{c*})_i > 0 \quad ((s_u^{c*})_i > 0) \quad (13.14)$$

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

$$(s_l^{x*})_j > 0 \quad ((s_u^{x*})_i > 0) \quad (13.15)$$

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

13.2.7 The certificate of dual infeasibility

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

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && \bar{l}^c \leq Ax \leq \bar{u}^c, \\ & && \bar{l}^x \leq x \leq \bar{u}^x \end{aligned} \quad (13.16)$$

with negative objective value, where we use the definitions

$$\bar{l}_i^c := \begin{cases} 0, & l_i^c > -\infty, \\ -\infty, & \text{otherwise,} \end{cases} \quad \bar{u}_i^c := \begin{cases} 0, & u_i^c < \infty, \\ \infty, & \text{otherwise,} \end{cases} \quad (13.17)$$

and

$$\bar{l}_i^x := \begin{cases} 0, & l_i^x > -\infty, \\ -\infty, & \text{otherwise,} \end{cases} \quad \text{and} \quad \bar{u}_i^x := \begin{cases} 0, & u_i^x < \infty, \\ \infty, & \text{otherwise.} \end{cases} \quad (13.18)$$

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

$$\begin{aligned} c^T x^* &< 0, \\ \bar{l}^c &\leq Ax^* \leq \bar{u}^c, \\ \bar{l}^x &\leq x^* \leq \bar{u}^x \end{aligned} \quad (13.19)$$

The well-known Farkas Lemma tells us that (13.9) is infeasible if and only if a certificate of dual infeasibility exists.

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

$$x_j^* \neq 0, \quad (13.20)$$

variable j is involved in the dual infeasibility.

Chapter 14

Sensitivity analysis

14.1 Introduction

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

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

14.2 Restrictions

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

14.3 References

The book [16] discusses the classical sensitivity analysis in Chapter 10 whereas the book [22, Chapter 19] presents a modern introduction to sensitivity analysis. Finally, it is recommended to read the short paper [24] to avoid some of the pitfalls associated with sensitivity analysis.

14.4 Sensitivity analysis for linear problems

14.4.1 The optimal objective value function

Assume that we are given the problem

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

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

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

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

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

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

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

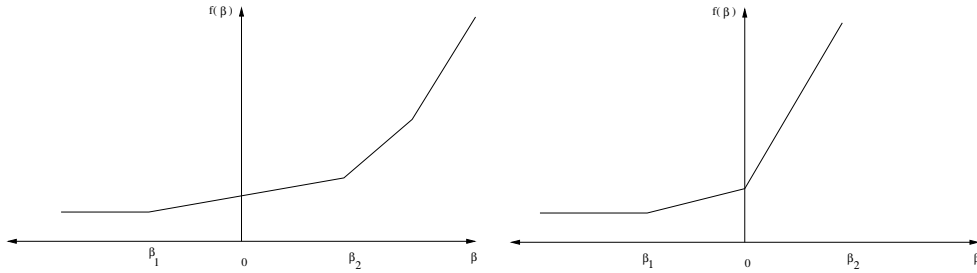


Figure 14.1: The optimal value function $f_{l_i^c}(\beta)$. Left: $\beta = 0$ is in the interior of linearity interval. Right: $\beta = 0$ is a breakpoint.

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

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

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

$$\beta \in [\beta_1, \beta_2] \quad (14.5)$$

for which

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

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

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

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

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

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

14.4.1.1 Equality constraints

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

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

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

14.4.2 The basis type sensitivity analysis

The classical sensitivity analysis discussed in most textbooks about linear optimization, e.g. [16, Chapter 10], is based on an optimal basic solution or equivalently on an optimal basis. This method may produce misleading results [22, Chapter 19] but is **computationally cheap**. Therefore, and for historical reasons this method is available in MOSEK.

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

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

14.4.3 The optimal partition type sensitivity analysis

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

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

$$z^* := c^T x^*. \quad (14.9)$$

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

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

and

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

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

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

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

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

and

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

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

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

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

and

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

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

$$x_j^* \in [\sigma_1, \sigma_2]. \quad (14.17)$$

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

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

and

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

14.4.4 An example

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

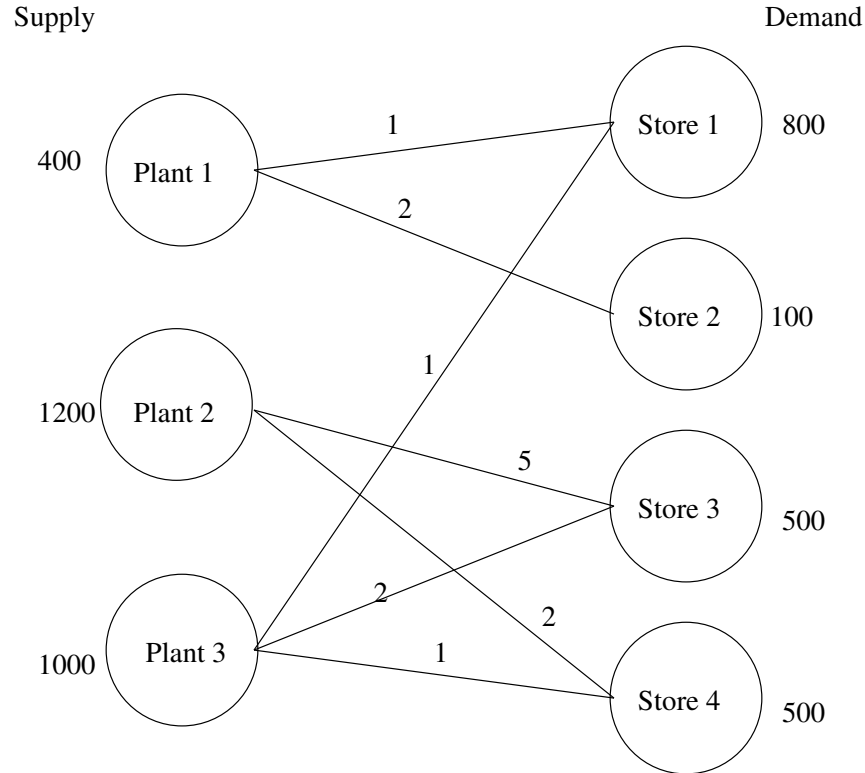


Figure 14.2: Supply, demand and cost of transportation.

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

minimize

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

subject to

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

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

Basis type					Optimal partition type				
Con.	β_1	β_2	σ_1	σ_2	Con.	β_1	β_2	σ_1	σ_2
1	-300.00	0.00	3.00	3.00	1	-300.00	500.00	3.00	1.00
2	-700.00	$+\infty$	0.00	0.00	2	-700.00	$+\infty$	-0.00	-0.00
3	-500.00	0.00	3.00	3.00	3	-500.00	500.00	3.00	1.00
4	-0.00	500.00	4.00	4.00	4	-500.00	500.00	2.00	4.00
5	-0.00	300.00	5.00	5.00	5	-100.00	300.00	3.00	5.00
6	-0.00	700.00	5.00	5.00	6	-500.00	700.00	3.00	5.00
7	-500.00	700.00	2.00	2.00	7	-500.00	700.00	2.00	2.00
Var.	β_1	β_2	σ_1	σ_2	Var.	β_1	β_2	σ_1	σ_2
x_{11}	$-\infty$	300.00	0.00	0.00	x_{11}	$-\infty$	300.00	0.00	0.00
x_{12}	$-\infty$	100.00	0.00	0.00	x_{12}	$-\infty$	100.00	0.00	0.00
x_{23}	$-\infty$	0.00	0.00	0.00	x_{23}	$-\infty$	500.00	0.00	2.00
x_{24}	$-\infty$	500.00	0.00	0.00	x_{24}	$-\infty$	500.00	0.00	0.00
x_{31}	$-\infty$	500.00	0.00	0.00	x_{31}	$-\infty$	500.00	0.00	0.00
x_{33}	$-\infty$	500.00	0.00	0.00	x_{33}	$-\infty$	500.00	0.00	0.00
x_{34}	-0.000000	500.00	2.00	2.00	x_{34}	$-\infty$	500.00	0.00	2.00

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

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

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

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

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

then the optimal objective value will decrease by the value

$$\sigma_2\beta = 1\beta. \quad (14.23)$$

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

$$\beta \in [0, 300] \quad (14.24)$$

then the optimal objective value will increased by the value

$$\sigma_1\beta = 3\beta. \quad (14.25)$$

14.5 Sensitivity analysis in the MATLAB toolbox

The following describe sensitivity analysis from the MATLAB toolbox.

14.5.1 On bounds

The index of bounds/variables to analyzed for sensitivity are specified in the following subfields of the matlab structure **prob**:

- `.prisen.cons.subu` Indexes of constraints, where upper bounds are analyzed for sensitivity.
- `.prisen.cons.subl` Indexes of constraints, where lower bounds are analyzed for sensitivity.
- `.prisen.vars.subu` Indexes of variables, where upper bounds are analyzed for sensitivity.
- `.prisen.vars.subl` Indexes of variables, where lower bounds are analyzed for sensitivity.

`.duasen.sub` Index of variables where coefficients are analysed for sensitivity.

For an equality constraint, the index can be specified in either `subu` or `subl`. After calling `[r,res] = mosekopt('minimize',prob)` the results are returned in the subfields `prisen` and `duasen` of `res`.

14.5.1.1 `prisen`

The field `prisen` is structured as follows:

<code>.cons</code>	MATLAB structure with subfields:
<code>.lr_bl</code>	Left value β_1 in the linearity interval for a lower bound.
<code>.rr_bl</code>	Right value β_2 in the linearity interval for a lower bound.
<code>.ls_bl</code>	Left shadow price s_l for a lower bound.
<code>.rs_bl</code>	Right shadow price s_r for a lower bound.
<code>.lr_bu</code>	Left value β_1 in the linearity interval for an upper bound.
<code>.rr_bu</code>	Right value β_2 in the linearity interval for an upper bound.
<code>.ls_bu</code>	Left shadow price s_l for an upper bound.
<code>.rs_bu</code>	Right shadow price s_r for an upper bound.
<code>.var</code>	MATLAB structure with subfields:
<code>.lr_bl</code>	Left value β_1 in the linearity interval for a lower bound on a variable.
<code>.rr_bl</code>	Right value β_2 in the linearity interval for a lower bound on a variable.
<code>.ls_bl</code>	Left shadow price s_l for a lower bound on a variable.
<code>.rs_bl</code>	Right shadow price s_r for lower bound on a variable.
<code>.lr_bu</code>	Left value β_1 in the linearity interval for an upper bound on a variable.
<code>.rr_bu</code>	Right value β_2 in the linearity interval for an upper bound on a variable.
<code>.ls_bu</code>	Left shadow price s_l for an upper bound on a variables.
<code>.rs_bu</code>	Right shadow price s_r for an upper bound on a variables.

14.5.1.2 duasen

The field `duasen` is structured as follows:

<code>.lr_c</code>	Left value β_1 of linearity interval for an objective coefficient.
<code>.rr_c</code>	Right value β_2 of linearity interval for an objective coefficient.
<code>.ls_c</code>	Left shadow price s_l for an objective coefficients .
<code>.rs_c</code>	Right shadow price s_r for an objective coefficients.

14.5.2 Selecting analysis type

The type (basis or optimal partition) of analysis to be performed can be selected by setting the parameter

`MSK_IPAR_SENSITIVITY_TYPE`

to one of the values:

```
MSK_SENSITIVITY_TYPE_BASIS = 0
MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION = 1
```

as seen in the following example.

14.5.3 An example

Consider the problem defined in (14.21). Suppose we wish to perform sensitivity analysis on all bounds and coefficients. The following example demonstrates this as well as the method for changing between basic and full sensitivity analysis.

```
% sensitivity.m

% Obtain all symbolic constants
% defined by MOSEK.
[r,res] = mosekopt('symbcon');
sc       = res.symbcon;
[r,res] = mosekopt('read(transport.lp) echo(0)');
prob = res.prob;
% analyse upper bound 1:7
prob.prisen.cons.subl = [];
prob.prisen.cons.subu = [1:7];
% analyse lower bound on variables 1:7
prob.prisen.vars.subl = [1:7];
```

```

prob.prisen.vars.subu = [];
% analyse coefficient 1:7
prob.duasen.sub = [1:7];
%Select basis sensitivity analysis and optimize.
param.MSK_IPAR_SENSITIVITY_TYPE=sc.MSK_SENSITIVITY_TYPE_BASIS;
[r,res] = mosekopt('minimize echo(0)',prob,param);
results(1) = res;
% Select optimal partition sensitivity analysis and optimize.
param.MSK_IPAR_SENSITIVITY_TYPE=sc.MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION;
[r,res] = mosekopt('minimize echo(0)',prob,param);
results(2) = res;
%Print results
for m = [1:2]
    if m == 1
        fprintf('\nBasis sensitivity results:\n')
    else
        fprintf('\nOptimal partition sensitivity results:\n')
    end
    fprintf('\nSensitivity for bounds on constraints:\n')
    for i = 1:length(prob.prisen.cons.subl)
        fprintf (...
            'con = %d, beta_1 = %.1f, beta_2 = %.1f, delta_1 = %.1f,delta_2 = %.1f\n', ...
            prob.prisen.cons.subu(i),results(m).prisen.cons.lr_bu(i), ...
            results(m).prisen.cons.rr_bu(i),...
            results(m).prisen.cons.ls_bu(i),...
            results(m).prisen.cons.rs_bu(i));
    end

    for i = 1:length(prob.prisen.cons.subu)
        fprintf (...
            'con = %d, beta_1 = %.1f, beta_2 = %.1f, delta_1 = %.1f,delta_2 = %.1f\n', ...
            prob.prisen.cons.subu(i),results(m).prisen.cons.lr_bu(i), ...
            results(m).prisen.cons.rr_bu(i),...
            results(m).prisen.cons.ls_bu(i),...
            results(m).prisen.cons.rs_bu(i));
    end
    fprintf('Sensitivity for bounds on variables:\n')
    for i = 1:length(prob.prisen.vars.subl)
        fprintf (...
            'var = %d, beta_1 = %.1f, beta_2 = %.1f, delta_1 = %.1f,delta_2 = %.1f\n', ...
            prob.prisen.vars.subl(i),results(m).prisen.vars.lr_bl(i), ...

```

```

    results(m).prisen.vars.rr_bl(i),...
    results(m).prisen.vars.ls_bl(i),...
    results(m).prisen.vars.rs_bl(i));
end

for i = 1:length(prob.prisen.vars.subu)
    fprintf (...
        'var = %d, beta_1 = %.1f, beta_2 = %.1f, delta_1 = %.1f,delta_2 = %.1f\n', ...
        prob.prisen.vars.subu(i),results(m).prisen.vars.lr_bu(i), ...
        results(m).prisen.vars.rr_bu(i),...
        results(m).prisen.vars.ls_bu(i),...
        results(m).prisen.vars.rs_bu(i));
    end
end

fprintf('Sensitivity for coefficients in objective:\n')
for i = 1:length(prob.duasen.sub)
    fprintf (...
        'var = %d, beta_1 = %.1f, beta_2 = %.1f, delta_1 = %.1f,delta_2 = %.1f\n', ...
        prob.duasen.sub(i),results(m).duasen.lr_c(i), ...
        results(m).duasen.rr_c(i),...
        results(m).duasen.ls_c(i),...
        results(m).duasen.rs_c(i));
    end
end
end

```

The output from running the example `sensitivity.m` is shown below.

Basis sensitivity results:

Sensitivity for bounds on constraints:

```

con = 1, beta_1 = -300.0, beta_2 = 0.0, delta_1 = 3.0,delta_2 = 3.0
con = 2, beta_1 = -700.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0
con = 3, beta_1 = -500.0, beta_2 = 0.0, delta_1 = 3.0,delta_2 = 3.0
con = 4, beta_1 = -0.0, beta_2 = 500.0, delta_1 = 4.0,delta_2 = 4.0
con = 5, beta_1 = -0.0, beta_2 = 300.0, delta_1 = 5.0,delta_2 = 5.0
con = 6, beta_1 = -0.0, beta_2 = 700.0, delta_1 = 5.0,delta_2 = 5.0
con = 7, beta_1 = -500.0, beta_2 = 700.0, delta_1 = 2.0,delta_2 = 2.0

```

Sensitivity for bounds on variables:

```

var = 1, beta_1 = Inf, beta_2 = 300.0, delta_1 = 0.0,delta_2 = 0.0
var = 2, beta_1 = Inf, beta_2 = 100.0, delta_1 = 0.0,delta_2 = 0.0
var = 3, beta_1 = Inf, beta_2 = 0.0, delta_1 = 0.0,delta_2 = 0.0

```

```

var = 4, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 5, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 6, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 7, beta_1 = -0.0, beta_2 = 500.0, delta_1 = 2.0,delta_2 = 2.0
Sensitivity for coefficients in objective:
var = 1, beta_1 = Inf, beta_2 = 3.0, delta_1 = 300.0,delta_2 = 300.0
var = 2, beta_1 = Inf, beta_2 = Inf, delta_1 = 100.0,delta_2 = 100.0
var = 3, beta_1 = -2.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0
var = 4, beta_1 = Inf, beta_2 = 2.0, delta_1 = 500.0,delta_2 = 500.0
var = 5, beta_1 = -3.0, beta_2 = Inf, delta_1 = 500.0,delta_2 = 500.0
var = 6, beta_1 = Inf, beta_2 = 2.0, delta_1 = 500.0,delta_2 = 500.0
var = 7, beta_1 = -2.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0

```

Optimal partition sensitivity results:

```

Sensitivity for bounds on constraints:
con = 1, beta_1 = -300.0, beta_2 = 500.0, delta_1 = 3.0,delta_2 = 1.0
con = 2, beta_1 = -700.0, beta_2 = Inf, delta_1 = -0.0,delta_2 = -0.0
con = 3, beta_1 = -500.0, beta_2 = 500.0, delta_1 = 3.0,delta_2 = 1.0
con = 4, beta_1 = -500.0, beta_2 = 500.0, delta_1 = 2.0,delta_2 = 4.0
con = 5, beta_1 = -100.0, beta_2 = 300.0, delta_1 = 3.0,delta_2 = 5.0
con = 6, beta_1 = -500.0, beta_2 = 700.0, delta_1 = 3.0,delta_2 = 5.0
con = 7, beta_1 = -500.0, beta_2 = 700.0, delta_1 = 2.0,delta_2 = 2.0
Sensitivity for bounds on variables:
var = 1, beta_1 = Inf, beta_2 = 300.0, delta_1 = 0.0,delta_2 = 0.0
var = 2, beta_1 = Inf, beta_2 = 100.0, delta_1 = 0.0,delta_2 = 0.0
var = 3, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 2.0
var = 4, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 5, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 6, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 7, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 2.0
Sensitivity for coefficients in objective:
var = 1, beta_1 = Inf, beta_2 = 3.0, delta_1 = 300.0,delta_2 = 300.0
var = 2, beta_1 = Inf, beta_2 = Inf, delta_1 = 100.0,delta_2 = 100.0
var = 3, beta_1 = -2.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0
var = 4, beta_1 = Inf, beta_2 = 2.0, delta_1 = 500.0,delta_2 = 500.0
var = 5, beta_1 = -3.0, beta_2 = Inf, delta_1 = 500.0,delta_2 = 500.0
var = 6, beta_1 = Inf, beta_2 = 2.0, delta_1 = 500.0,delta_2 = 500.0
var = 7, beta_1 = -2.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0

```


Appendix A

The MPS file format

MOSEK supports the standard MPS format with some extensions. For a detailed description of the MPS format the book by Nazareth [21] is a good reference.

A.1 The MPS file format

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

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

where

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

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

where it is assumed that

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

Please note the explicit $1/2$ in the quadratic term and that Q^i is required to be symmetric.

- \mathcal{C} is a convex cone.
- $\mathcal{J} \subseteq \{1, 2, \dots, n\}$ 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
    [vname1]  [vname2]    [value1]    [vname3]  [value2]
BOUNDS
    ?? [name]  [vname1]    [value1]
CSECTION
    [vname1]  [kname1]    [value1]    [ktype]
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

$$[+|-]XXXXXXXX.XXXXXX[[e|E][+|-]XXX]$$

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 [A.5](#) for details.

A.1.1 An example

A concrete example of a MPS file is presented below:

```

NAME          EXAMPLE
OBJSENSE
  MIN
ROWS
  N  obj
  L  c1
  L  c2
  L  c3
  L  c4
COLUMNS
  x1      obj      -10.0      c1      0.7
  x1      c2        0.5      c3      1.0
  x1      c4        0.1
  x2      obj      -9.0      c1      1.0
  x2      c2      0.833333333333 c3      0.666666667
  x2      c4        0.25
RHS
  rhs      c1      630.0      c2      600.0
  rhs      c3      708.0      c4      135.0
ENDATA

```

Subsequently each individual section in the MPS format is discussed.

A.1.2 NAME

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

A.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.

A.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.

A.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 .

A.1.6 COLUMNS

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

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

where the requirements for each field are as follows:

Field	Starting position	Maximum width	Required	Description
[vname1]	5	8	Yes	Variable name
[cname1]	15	8	Yes	Constraint name
[value1]	25	12	Yes	Numerical value
[cname2]	40	8	No	Constraint name
[value2]	50	12	No	Numerical value

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

- All elements belonging to one variable must be grouped together.
- Zero elements of A should not be specified.
- At least one element for each variable should be specified.

A.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.

A.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	Re-quired	Description
[name]	5	8	Yes	Name of the RANGE vector
[cname1]	15	8	Yes	Constraint name
[value1]	25	12	Yes	Numerical value
[cname2]	40	8	No	Constraint name
[value2]	50	12	No	Numerical value

The records in this section are used to modify the bound vectors for the constraints, i.e. the values in l^c and u^c . A record has the following interpretation: [name] is the name of the

RANGE vector anhd [cname1] is a valid constraint name. Assume that [cname1] is assigned to the 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			

A.1.9 QSECTION (optional)

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

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

where the requirements for each field are:

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

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

The example

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

has the following MPS file representation

```

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

```

Regarding the QSECTIONs please note that:

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

A.1.10 BOUNDS (optional)

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

```
?? [name]      [vname1]      [value1]
```

where the requirements for each field are:

Field	Starting position	Maximum width	Required	Description
??	2	2	Yes	Bound key
[name]	5	8	Yes	Name of the BOUNDS vector
[vname1]	15	8	Yes	Variable name
[value1]	25	12	No	Variable name

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

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

v_1 is the value specified by [value1].

A.1.11 CSECTION (optional)

The purpose of the CSECTION is to specify the constraint

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

in (A.1).

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

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

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

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

Next define

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

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:

$$\mathcal{C}_t = \left\{ x \in \mathbb{R}^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\}. \quad (\text{A.3})$$

- Rotated quadratic cone:

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

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

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

$$x_4 \geq \sqrt{x_5^2 + x_0^2} \quad (\text{A.5})$$

and the rotated quadratic cone

$$2x_3x_7 \geq x_1^2 + x_8^2, x_3, x_7 \geq 0, \quad (\text{A.6})$$

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

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

This first CSECTION specifies the cone (A.5) which is given the name **konea**. This is a quadratic cone which is specified by the keyword **QUAD** in the CSECTION header. The 0.0 value in the CSECTION header is not used by the **QUAD** cone.

The second CSECTION specifies the rotated quadratic cone (A.6). Please note the keyword **RQUAD** in the CSECTION which is used to specify that the cone is a rotated quadratic cone instead of a quadratic cone. The 0.0 value in the CSECTION header is not used by the **RQUAD** cone.

In general, a CSECTION header has the format

CSECTION [kname1] [value1] [ktype]

where the requirement for each field are as follows:

Field	Starting position	Maximum width	Required	Description
[kname1]	5	8	Yes	Name of the cone
[value1]	15	12	No	Cone parameter
[ktype]	25		Yes	Type of the cone.

The possible cone type keys are:

Cone type key	Members	Interpretation.
QUAD	≥ 1	Quadratic cone i.e. (A.3).
RQUAD	≥ 2	Rotated quadratic cone i.e. (A.4).

Please note that a quadratic cone must have at least one member whereas a rotated quadratic cone must have at least two members. A record in the CSECTION has the format

[vname1]

where the requirements for each field are

Field	Starting position	Maximum width	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.

A.1.12 ENDATA

This keyword denotes the end of the MPS file.

A.2 Integer variables

Using special bound keys in the `BOUNDS` section it is possible to specify that some or all of the variables should be integer-constrained i.e. be members of \mathcal{J} . However, an alternative method is available.

This method is available only for backward compability and we recommend that it is not used. This method requires that markers are placed in the `COLUMNS` section as in the example:

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

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

- **IMPORTANT:** All variables between the markers are assigned a default lower bound of 0 and a default upper bound of 1. **This may not be what is intended.** If it is not intended, the correct bounds should be defined in the `BOUNDS` section of the MPS formatted file.
- MOSEK ignores field 1, i.e. `MARK0001` and `MARK001`, however, other optimization systems require them.
- Field 2, i.e. `'MARKER'`, must be specified including the single quotes. This implies that no row can be assigned the name `'MARKER'`.
- Field 3 is ignored and should be left blank.
- Field 4, i.e. `'INTORG'` and `'INTEND'`, must be specified.
- It is possible to specify several such integer marker sections within the `COLUMNS` section.

A.3 General limitations

- An MPS file should be an ASCII file.

A.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.

A.5 The free MPS format

MOSEK supports a free format variation of the MPS format. The free format is similar to the MPS file format but less restrictive, e.g. it allows longer names. However, it also presents two main limitations:

- By default a line in the MPS file must not contain more than 1024 characters. However, by modifying the parameter `MSK_IPAR_READ_MPS_WIDTH` an arbitrary large line width will be accepted.
- A name must not contain any blanks.

To use the free MPS format instead of the default MPS format the MOSEK parameter `MSK_IPAR_READ_MPS_FORMAT` should be changed.

Appendix B

The LP file format

MOSEK supports the LP file format with some extensions i.e. MOSEK can read and write LP formatted files.

B.1 A warning

The LP format is not a well-defined standard and hence different optimization packages may interpretate a specific LP formatted file differently.

B.2 The LP file format

The LP file format can specify problems on the form

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

where

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

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

and it is assumed that

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

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

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

where it is assumed that

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

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

B.2.1 The sections

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

B.2.1.1 The objective

The first section beginning with one of the keywords

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

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

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

The objective may be given a name by writing

```
myname:
```

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

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

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

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

```
x1 ^ 2
```

and

```
x1 * x2
```

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

An example of an objective section is:

```
minimize
```

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

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

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

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

B.2.1.2 The constraints

The second section beginning with one of the keywords

```
subj to
```

```
subject to
```

```
s.t.
```

```
st
```

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

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

```
subject to
```

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

The bound type (here \leq) may be any of $<$, \leq , $=$, $>$, \geq ($<$ and \leq 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.

$$-5 \leq x_1 + x_2 \leq 5 \tag{B.3}$$

may be written as

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

By default MOSEK writes ranged constraints this way.

If the files must adhere to the LP standard, ranged constraints must either be split into upper bounded and lower bounded constraints or be written as an equality with a slack variable. For example the expression (B.3) may be written as

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

B.2.1.3 Bounds

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

```
bound
bounds
```

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

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

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

B.2.1.4 Variable types

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

```
bin
binaries
binary
```

and

```
gen
general
```

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

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

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

B.2.1.5 Terminating section

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

```
end
```

B.2.1.6 An example

A simple example of an LP file with two variables, four constraints and one integer variable is:

```
minimize
  -10 x1 -9 x2
subject to
  0.7 x1 +      x2 <= 630
  0.5 x1 + 0.833 x2 <= 600
      x1 + 0.667 x2 <= 708
  0.1 x1 + 0.025 x2 <= 135
bounds
  10 <= x1
  x1 <= +inf
```

```

    20 <= x2 <= 500
general
    x1
end

```

B.2.2 LP format peculiarities

B.2.2.1 Comments

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

B.2.2.2 Names

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

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

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

It is strongly recommended not to use double quotes (") in names.

B.2.2.3 Variable bounds

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

B.2.2.4 MOSEK specific extensions to the LP format

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

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

If an LP formatted file created by MOSEK should satisfy 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 lose 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.

B.2.3 The strict LP format

The LP format is not a formal standard and different vendors have slightly different interpretations of the LP format. To make MOSEK's definition of the LP format more compatible 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.

B.2.4 Formatting of an LP file

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

```
MSK_IPAR_WRITE_LP_LINE_WIDTH
```

```
MSK_IPAR_WRITE_LP_TERMS_PER_LINE
```

The first parameter sets the maximum number of characters on a single line. The default value is 80 corresponding roughly to the width of a standard text document.

The second parameter sets the maximum number of terms per line; a term means a sign, a coefficient, and a name (for example “+ 42 elephants”). The default value is 0, meaning that there is no maximum.

B.2.4.1 Speeding up file reading

If the input file should be read as fast as possible using the least amount of memory, then it is important to tell MOSEK how many non-zeros, variables and constraints the problem contains. These values can be set using the parameters

```
MSK_IPAR_READ_CON  
MSK_IPAR_READ_VAR  
MSK_IPAR_READ_ANZ  
MSK_IPAR_READ_QNZ
```

B.2.4.2 Unnamed constraints

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

Appendix C

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.

C.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.

C.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]
```

C.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

$$-10 \leq x, y \leq 10. \quad (\text{C.1})$$

- **[cone]**. Currently, the supported cones are the *quadratic cone* and the *rotated quadratic cone*. A conic constraint is defined as a set of variables which belongs to a single unique cone.

A quadratic cone of n variables x_1, \dots, 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, \dots, 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 number of full or partial solutions to a problem, each inside a **[solution]**-section. The syntax is

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

where **SOLTYPE** is one of the strings

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

and **STATUS** is one of the strings

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

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

A **[solution]**-section contains **[con]** and **[var]** sections. Each **[con]** and **[var]** section defines solution values for a single variable or constraint, each value written as

```
KEYWORD=value
```

where **KEYWORD** defines a solution item and **value** defines its value. Allowed keywords are as follows:

- **sk**. The status of the item, where the **value** is one of the following strings:
 - * **LOW**, the item is on its lower bound.
 - * **UPR**, the item is on its upper bound.
 - * **FIX**, it is a fixed item.
 - * **BAS**, the item is in the basis.
 - * **SUPBAS**, the item is super basic.
 - * **UNK**, the status is unknown.
 - * **INF**, the item is outside its bounds (infeasible).
- **lvl** Defines the level of the item.
- **s1** Defines the level of the variable associated with its lower bound.
- **su** Defines the level of the variable associated with its upper bound.
- **sn** Defines the level of the variable associated with its cone.
- **y** Defines the level of the corresponding dual variable (for constraints only).

A **[var]** section should always contain the items **sk** and **lvl**, and optionally **s1**, **su** and **sn**.

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

An example of a solution section

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

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

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

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

C.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]+)?
```

C.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"
```

C.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]
```

C.4 Writing OPF files from MOSEK

To write an OPF file set the parameter `MSK_IPAR_WRITE_DATA_FORMAT` to `MSK_DATA_FORMAT_OPF` 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.

C.5 Examples

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

C.5.1 Linear example lo1.opf

Consider the example:

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

In the OPF format the example is displayed as shown below:

```

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

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

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

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

[constraints]
  [con 'c1'] 0.7 x1 + x2 <= 630 [/con]
  [con 'c2'] 0.5 x1 + 0.8333333333 x2 <= 600 [/con]
  [con 'c3'] x1 + 0.666666667 x2 <= 708 [/con]
  [con 'c4'] 0.1 x1 + 0.25 x2 <= 135 [/con]
[/constraints]

[bounds]
  # By default all variables are free. The following line will

```

```
# change this to all variables being nonnegative.
[b] 0 <= * [/b]
[/bounds]
```

C.5.2 Quadratic example qo1.opf

An example of a quadratic optimization problem is

$$\begin{aligned} & \text{minimize} && x_1^2 + 0.1x_2^2 + x_3^2 - x_1x_3 - x_2 \\ & \text{subject to} && 1 \leq x_1 + x_2 + x_3, \\ & && x \geq 0. \end{aligned} \tag{C.3}$$

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

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

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

[variables]
  x1 x2 x3
[/variables]

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

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

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

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

C.5.3 Conic quadratic example `cqo1.opf`

Consider the example:

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

Please note that the type of the cones is defined by the parameter to `[cone ...]`; the content of the `cone`-section is the names of variables that belong to the cone.

```

[comment]
  Example cqo1.mps converted to OPF.
[/comment]

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

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

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

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

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

  # We define two rotated quadratic cones

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

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

```

```
[/bounds]
```

C.5.4 Mixed integer example milo1.opf

Consider the mixed integer problem:

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

This can be implemented in OPF with:

```
[comment]
  Written by MOSEK version 5.0.0.7
  Date 20-11-06
  Time 14:42:24
[/comment]

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

[variables disallow_new_variables]
  x1 x2
[/variables]

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

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

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

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

Appendix D

The XML (OSiL) format

MOSEK can write data in the standard OSiL xml format. For a definition of the OSiL format please see <http://www.optimizationservices.org/>. Only linear constraints (possibly with integer variables) are supported. By default output files with the extension `.xml` are written in the OSiL format.

The parameter `MSK_IPAR_WRITE_XML_MODE` controls if the linear coefficients in the A matrix are written in row or column order.

Appendix E

Parameters

Subsequently all parameters that are in MOSEK parameter database is presented. For each parameter their name, purpose, type, default value etc. are presented.

E.1 Parameter groups

Parameters grouped by meaning and functionality.

E.1.1 Logging parameters.

- **MSK_IPAR_LOG** 322
Controls the amount of log information.
- **MSK_IPAR_LOG_BI** 322
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** 323
Controls the logging frequency.
- **MSK_IPAR_LOG_CONCURRENT** 323
Controls amount of output printed by the concurrent optimizer.
- **MSK_IPAR_LOG_CUT_SECOND_OPT** 324
Controls the reduction in the log levels for the second and any subsequent optimizations.
- **MSK_IPAR_LOG_FACTOR** 324
If turned on, then the factor log lines are added to the log.
- **MSK_IPAR_LOG_FEASREPAIR** 324
Controls the amount of output printed when performing feasibility repair.

- **MSK_IPAR_LOG_FILE** 325
If turned on, then some log info is printed when a file is written or read.
- **MSK_IPAR_LOG_HEAD** 325
If turned on, then a header line is added to the log.
- **MSK_IPAR_LOG_INFEAS_ANA** 325
Controls log level for the infeasibility analyzer.
- **MSK_IPAR_LOG_INTPNT** 326
Controls the amount of log information from the interior-point optimizers.
- **MSK_IPAR_LOG_MIO** 326
Controls the amount of log information from the mixed-integer optimizers.
- **MSK_IPAR_LOG_MIO_FREQ** 326
The mixed-integer solver logging frequency.
- **MSK_IPAR_LOG_NONCONVEX** 326
Controls amount of output printed by the nonconvex optimizer.
- **MSK_IPAR_LOG_OPTIMIZER** 327
Controls the amount of general optimizer information that is logged.
- **MSK_IPAR_LOG_ORDER** 327
If turned on, then factor lines are added to the log.
- **MSK_IPAR_LOG_PARAM** 327
Controls the amount of information printed out about parameter changes.
- **MSK_IPAR_LOG_PRESOLVE** 328
Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.
- **MSK_IPAR_LOG_RESPONSE** 328
Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.
- **MSK_IPAR_LOG_SENSITIVITY** 328
Control logging in sensitivity analyzer.
- **MSK_IPAR_LOG_SENSITIVITY_OPT** 328
Control logging in sensitivity analyzer.
- **MSK_IPAR_LOG_SIM** 329
Controls the amount of log information from the simplex optimizers.

- **MSK_IPAR_LOG_SIM_FREQ** 329
Controls simplex logging frequency.
- **MSK_IPAR_LOG_SIM_NETWORK_FREQ** 330
Controls the network simplex logging frequency.
- **MSK_IPAR_LOG_STORAGE** 330
Controls the memory related log information.

E.1.2 Basis identification parameters.

- **MSK_IPAR_BI_CLEAN_OPTIMIZER** 307
Controls which simplex optimizer is used in the clean-up phase.
- **MSK_IPAR_BI_IGNORE_MAX_ITER** 308
Turns on basis identification in case the interior-point optimizer is terminated due to maximum number of iterations.
- **MSK_IPAR_BI_IGNORE_NUM_ERROR** 308
Turns on basis identification in case the interior-point optimizer is terminated due to a numerical problem.
- **MSK_IPAR_BI_MAX_ITERATIONS** 308
Maximum number of iterations after basis identification.
- **MSK_IPAR_INTPNT_BASIS** 314
Controls whether basis identification is performed.
- **MSK_IPAR_LOG_BI** 322
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** 323
Controls the logging frequency.
- **MSK_DPAR_SIM_LU_TOL_REL_PIV** 290
Relative pivot tolerance employed when computing the LU factorization of the basis matrix.

E.1.3 The Interior-point method parameters.

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

- **MSK_IPAR_BI_IGNORE_MAX_ITER** 308
Turns on basis identification in case the interior-point optimizer is terminated due to maximum number of iterations.
- **MSK_IPAR_BI_IGNORE_NUM_ERROR** 308
Turns on basis identification in case the interior-point optimizer is terminated due to a numerical problem.
- **MSK_DPAR_CHECK_CONVEXITY_REL_TOL** 272
Convexity check tolerance.
- **MSK_IPAR_INTPNT_BASIS** 314
Controls whether basis identification is performed.
- **MSK_DPAR_INTPNT_CO_TOL_DFEAS** 275
Dual feasibility tolerance used by the conic interior-point optimizer.
- **MSK_DPAR_INTPNT_CO_TOL_INFEAS** 276
Infeasibility tolerance for the conic solver.
- **MSK_DPAR_INTPNT_CO_TOL_MU_RED** 276
Optimality tolerance for the conic solver.
- **MSK_DPAR_INTPNT_CO_TOL_NEAR_REL** 276
Optimality tolerance for the conic solver.
- **MSK_DPAR_INTPNT_CO_TOL_PFEAS** 277
Primal feasibility tolerance used by the conic interior-point optimizer.
- **MSK_DPAR_INTPNT_CO_TOL_REL_GAP** 277
Relative gap termination tolerance used by the conic interior-point optimizer.
- **MSK_IPAR_INTPNT_DIFF_STEP** 315
Controls whether different step sizes are allowed in the primal and dual space.
- **MSK_IPAR_INTPNT_MAX_ITERATIONS** 316
Controls the maximum number of iterations allowed in the interior-point optimizer.
- **MSK_IPAR_INTPNT_MAX_NUM_COR** 316
Maximum number of correction steps.
- **MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS** 316
Maximum number of steps to be used by the iterative search direction refinement.
- **MSK_DPAR_INTPNT_NL_MERIT_BAL** 277
Controls if the complementarity and infeasibility is converging to zero at about equal rates.

- **MSK_DPAR_INTPNT_NL_TOL_DFEAS** 277
Dual feasibility tolerance used when a nonlinear model is solved.
- **MSK_DPAR_INTPNT_NL_TOL_MU_RED** 278
Relative complementarity gap tolerance.
- **MSK_DPAR_INTPNT_NL_TOL_NEAR_REL** 278
Nonlinear solver optimality tolerance parameter.
- **MSK_DPAR_INTPNT_NL_TOL_PFEAS** 278
Primal feasibility tolerance used when a nonlinear model is solved.
- **MSK_DPAR_INTPNT_NL_TOL_REL_GAP** 279
Relative gap termination tolerance for nonlinear problems.
- **MSK_DPAR_INTPNT_NL_TOL_REL_STEP** 279
Relative step size to the boundary for general nonlinear optimization problems.
- **MSK_IPAR_INTPNT_OFF_COL_TRH** 317
Controls the aggressiveness of the offending column detection.
- **MSK_IPAR_INTPNT_ORDER_METHOD** 317
Controls the ordering strategy.
- **MSK_IPAR_INTPNT_REGULARIZATION_USE** 318
Controls whether regularization is allowed.
- **MSK_IPAR_INTPNT_SCALING** 318
Controls how the problem is scaled before the interior-point optimizer is used.
- **MSK_IPAR_INTPNT_SOLVE_FORM** 319
Controls whether the primal or the dual problem is solved.
- **MSK_IPAR_INTPNT_STARTING_POINT** 319
Starting point used by the interior-point optimizer.
- **MSK_DPAR_INTPNT_TOL_DFEAS** 279
Dual feasibility tolerance used for linear and quadratic optimization problems.
- **MSK_DPAR_INTPNT_TOL_DSAFE** 279
Controls the interior-point dual starting point.
- **MSK_DPAR_INTPNT_TOL_INFEAS** 280
Nonlinear solver infeasibility tolerance parameter.
- **MSK_DPAR_INTPNT_TOL_MU_RED** 280
Relative complementarity gap tolerance.

- **MSK_DPAR_INTPNT_TOL_PATH** 280
interior-point centering aggressiveness.
- **MSK_DPAR_INTPNT_TOL_PFEAS** 281
Primal feasibility tolerance used for linear and quadratic optimization problems.
- **MSK_DPAR_INTPNT_TOL_PSAFE** 281
Controls the interior-point primal starting point.
- **MSK_DPAR_INTPNT_TOL_REL_GAP** 281
Relative gap termination tolerance.
- **MSK_DPAR_INTPNT_TOL_REL_STEP** 281
Relative step size to the boundary for linear and quadratic optimization problems.
- **MSK_DPAR_INTPNT_TOL_STEP_SIZE** 282
If the step size falls below the value of this parameter, then the interior-point optimizer assumes that it is stalled. It it does not not make any progress.
- **MSK_IPAR_LOG_CONCURRENT** 323
Controls amount of output printed by the concurrent optimizer.
- **MSK_IPAR_LOG_INTPNT** 326
Controls the amount of log information from the interior-point optimizers.
- **MSK_IPAR_LOG_PRESOLVE** 328
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** 289
This parameter determines when columns are dropped in incomplete cholesky factorization doing reformulation of quadratic problems.
- **MSK_IPAR_QO_SEPARABLE_REFORMULATION** 347
Determine if Quadratic programing problems should be reformulated to separable form.

E.1.4 Simplex optimizer parameters.

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

- **MSK_DPAR_BASIS_REL_TOL_S** 271
Maximum relative dual bound violation allowed in an optimal basic solution.
- **MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE** 307
Controls the sign of the columns in the basis matrix corresponding to slack variables.

- **MSK_DPAR_BASIS_TOL_S** 271
Maximum absolute dual bound violation in an optimal basic solution.
- **MSK_DPAR_BASIS_TOL_X** 271
Maximum absolute primal bound violation allowed in an optimal basic solution.
- **MSK_IPAR_LOG_SIM** 329
Controls the amount of log information from the simplex optimizers.
- **MSK_IPAR_LOG_SIM_FREQ** 329
Controls simplex logging frequency.
- **MSK_IPAR_LOG_SIM_MINOR** 329
Currently not in use.
- **MSK_IPAR_SENSITIVITY_OPTIMIZER** 354
Controls which optimizer is used for optimal partition sensitivity analysis.
- **MSK_IPAR_SIM_BASIS_FACTOR_USE** 355
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 penalty.
- **MSK_IPAR_SIM_DEGEN** 356
Controls how aggressively degeneration is handled.
- **MSK_IPAR_SIM_DUAL_PHASEONE_METHOD** 356
An experimental feature.
- **MSK_IPAR_SIM_EXPLOIT_DUPVEC** 358
Controls if the simplex optimizers are allowed to exploit duplicated columns.
- **MSK_IPAR_SIM_HOTSTART** 358
Controls the type of hot-start that the simplex optimizer perform.
- **MSK_IPAR_SIM_INTEGER** 359
An experimental feature.
- **MSK_DPAR_SIM_LU_TOL_REL_PIV** 290
Relative pivot tolerance employed when computing the LU factorization of the basis matrix.
- **MSK_IPAR_SIM_MAX_ITERATIONS** 359
Maximum number of iterations that can be used by a simplex optimizer.

- **MSK_IPAR_SIM_MAX_NUM_SETBACKS** 359
Controls how many set-backs that are allowed within a simplex optimizer.
- **MSK_IPAR_SIM_NETWORK_DETECT_METHOD** 360
Controls which type of detection method the network extraction should use.
- **MSK_IPAR_SIM_NON_SINGULAR** 361
Controls if the simplex optimizer ensures a non-singular basis, if possible.
- **MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD** 361
An experimental feature.
- **MSK_IPAR_SIM_REFORMULATION** 363
Controls if the simplex optimizers are allowed to reformulate the problem.
- **MSK_IPAR_SIM_SAVE_LU** 363
Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.
- **MSK_IPAR_SIM_SCALING** 364
Controls how much effort is used in scaling the problem before a simplex optimizer is used.
- **MSK_IPAR_SIM_SCALING_METHOD** 364
Controls how the problem is scaled before a simplex optimizer is used.
- **MSK_IPAR_SIM_SOLVE_FORM** 364
Controls whether the primal or the dual problem is solved by the primal-/dual- simplex optimizer.
- **MSK_IPAR_SIM_STABILITY_PRIORITY** 365
Controls how high priority the numerical stability should be given.
- **MSK_IPAR_SIM_SWITCH_OPTIMIZER** 365
Controls the simplex behavior.
- **MSK_DPAR_SIMPLEX_ABS_TOL_PIV** 290
Absolute pivot tolerance employed by the simplex optimizers.

E.1.5 Primal simplex optimizer parameters.

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

- **MSK_IPAR_SIM_PRIMAL_CRASH** 361
Controls the simplex crash.

- **MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION** 362
Controls how aggressively restricted selection is used.
- **MSK_IPAR_SIM_PRIMAL_SELECTION** 362
Controls the primal simplex strategy.

E.1.6 Dual simplex optimizer parameters.

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

- **MSK_IPAR_SIM_DUAL_CRASH** 356
Controls whether crashing is performed in the dual simplex optimizer.
- **MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION** 357
Controls how aggressively restricted selection is used.
- **MSK_IPAR_SIM_DUAL_SELECTION** 357
Controls the dual simplex strategy.

E.1.7 Network simplex optimizer parameters.

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

- **MSK_IPAR_LOG_SIM_NETWORK_FREQ** 330
Controls the network simplex logging frequency.
- **MSK_IPAR_SIM_NETWORK_DETECT** 360
Level of aggressiveness of network detection.
- **MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART** 360
Level of aggressiveness of network detection in a simplex hot-start.
- **MSK_IPAR_SIM_REFACTOR_FREQ** 363
Controls the basis refactoring frequency.

E.1.8 Nonlinear convex method parameters.

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

- **MSK_IPAR_CHECK_CONVEXITY** 310
Specify the level of convexity check on quadratic problems
- **MSK_DPAR_INTPNT_NL_MERIT_BAL** 277
Controls if the complementarity and infeasibility is converging to zero at about equal rates.

- **MSK_DPAR_INTPNT_NL_TOL_DFEAS** 277
Dual feasibility tolerance used when a nonlinear model is solved.
- **MSK_DPAR_INTPNT_NL_TOL_MU_RED** 278
Relative complementarity gap tolerance.
- **MSK_DPAR_INTPNT_NL_TOL_NEAR_REL** 278
Nonlinear solver optimality tolerance parameter.
- **MSK_DPAR_INTPNT_NL_TOL_PFEAS** 278
Primal feasibility tolerance used when a nonlinear model is solved.
- **MSK_DPAR_INTPNT_NL_TOL_REL_GAP** 279
Relative gap termination tolerance for nonlinear problems.
- **MSK_DPAR_INTPNT_NL_TOL_REL_STEP** 279
Relative step size to the boundary for general nonlinear optimization problems.
- **MSK_DPAR_INTPNT_TOL_INFEAS** 280
Nonlinear solver infeasibility tolerance parameter.
- **MSK_IPAR_LOG_CHECK_CONVEXITY** 323
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.

E.1.9 The conic interior-point method parameters.

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

- **MSK_DPAR_INTPNT_CO_TOL_DFEAS** 275
Dual feasibility tolerance used by the conic interior-point optimizer.
- **MSK_DPAR_INTPNT_CO_TOL_INFEAS** 276
Infeasibility tolerance for the conic solver.
- **MSK_DPAR_INTPNT_CO_TOL_MU_RED** 276
Optimality tolerance for the conic solver.
- **MSK_DPAR_INTPNT_CO_TOL_NEAR_REL** 276
Optimality tolerance for the conic solver.

- **MSK_DPAR_INTPNT_CO_TOL_PFEAS** 277
Primal feasibility tolerance used by the conic interior-point optimizer.
- **MSK_DPAR_INTPNT_CO_TOL_REL_GAP** 277
Relative gap termination tolerance used by the conic interior-point optimizer.

E.1.10 The mixed-integer optimization parameters.

- **MSK_IPAR_LOG_MIO** 326
Controls the amount of log information from the mixed-integer optimizers.
- **MSK_IPAR_LOG_MIO_FREQ** 326
The mixed-integer solver logging frequency.
- **MSK_IPAR_MIO_BRANCH_DIR** 331
Controls whether the mixed-integer optimizer is branching up or down by default.
- **MSK_IPAR_MIO_BRANCH_PRIORITIES_USE** 331
Controls whether branching priorities are used by the mixed-integer optimizer.
- **MSK_IPAR_MIO_CONSTRUCT_SOL** 332
Controls if an initial mixed integer solution should be constructed from the values of the integer variables.
- **MSK_IPAR_MIO_CONT_SOL** 332
Controls the meaning of interior-point and basic solutions in mixed integer problems.
- **MSK_IPAR_MIO_CUT_LEVEL_ROOT** 333
Controls the cut level employed by the mixed-integer optimizer at the root node.
- **MSK_IPAR_MIO_CUT_LEVEL_TREE** 333
Controls the cut level employed by the mixed-integer optimizer in the tree.
- **MSK_DPAR_MIO_DISABLE_TERM_TIME** 283
Certain termination criteria is disabled within the mixed-integer optimizer for period time specified by the parameter.
- **MSK_IPAR_MIO_FEASPUMP_LEVEL** 333
Controls the feasibility pump heuristic which is used to construct a good initial feasible solution.
- **MSK_IPAR_MIO_HEURISTIC_LEVEL** 334
Controls the heuristic employed by the mixed-integer optimizer to locate an initial integer feasible solution.

- **MSK_DPAR_MIO_HEURISTIC_TIME** 283
Time limit for the mixed-integer heuristics.
- **MSK_IPAR_MIO_HOTSTART** 334
Controls whether the integer optimizer is hot-started.
- **MSK_IPAR_MIO_KEEP_BASIS** 334
Controls whether the integer presolve keeps bases in memory.
- **MSK_IPAR_MIO_MAX_NUM_BRANCHES** 335
Maximum number of branches allowed during the branch and bound search.
- **MSK_IPAR_MIO_MAX_NUM_RELAXS** 335
Maximum number of relaxations in branch and bound search.
- **MSK_IPAR_MIO_MAX_NUM_SOLUTIONS** 336
Controls how many feasible solutions the mixed-integer optimizer investigates.
- **MSK_DPAR_MIO_MAX_TIME** 284
Time limit for the mixed-integer optimizer.
- **MSK_DPAR_MIO_MAX_TIME_APRX_OPT** 284
Time limit for the mixed-integer optimizer.
- **MSK_DPAR_MIO_NEAR_TOL_ABS_GAP** 284
Relaxed absolute optimality tolerance employed by the mixed-integer optimizer.
- **MSK_DPAR_MIO_NEAR_TOL_REL_GAP** 285
The mixed-integer optimizer is terminated when this tolerance is satisfied.
- **MSK_IPAR_MIO_NODE_OPTIMIZER** 337
Controls which optimizer is employed at the non-root nodes in the mixed-integer optimizer.
- **MSK_IPAR_MIO_NODE_SELECTION** 337
Controls the node selection strategy employed by the mixed-integer optimizer.
- **MSK_IPAR_MIO_OPTIMIZER_MODE** 338
An experimental feature.
- **MSK_IPAR_MIO_PRESOLVE_AGGREGATE** 338
Controls whether problem aggregation is performed in the mixed-integer presolve.
- **MSK_IPAR_MIO_PRESOLVE_PROBING** 338
Controls whether probing is employed by the mixed-integer presolve.

- **MSK_IPAR_MIO_PRESOLVE_USE** 339
Controls whether presolve is performed by the mixed-integer optimizer.
- **MSK_DPAR_MIO_REL_ADD_CUT_LIMITED** 285
Controls cut generation for mixed-integer optimizer.
- **MSK_DPAR_MIO_REL_GAP_CONST** 285
This value is used to compute the relative gap for the solution to an integer optimization problem.
- **MSK_IPAR_MIO_ROOT_OPTIMIZER** 339
Controls which optimizer is employed at the root node in the mixed-integer optimizer.
- **MSK_IPAR_MIO_STRONG_BRANCH** 340
The depth from the root in which strong branching is employed.
- **MSK_DPAR_MIO_TOL_ABS_GAP** 286
Absolute optimality tolerance employed by the mixed-integer optimizer.
- **MSK_DPAR_MIO_TOL_ABS_RELAX_INT** 286
Integer constraint tolerance.
- **MSK_DPAR_MIO_TOL_FEAS** 286
Feasibility tolerance for mixed integer solver. Any solution with maximum infeasibility below this value will be considered feasible.
- **MSK_DPAR_MIO_TOL_REL_GAP** 287
Relative optimality tolerance employed by the mixed-integer optimizer.
- **MSK_DPAR_MIO_TOL_REL_RELAX_INT** 287
Integer constraint tolerance.
- **MSK_DPAR_MIO_TOL_X** 287
Absolute solution tolerance used in mixed-integer optimizer.

E.1.11 Presolve parameters.

- **MSK_IPAR_PRESOLVE_ELIM_FILL** 345
Maximum amount of fill-in in the elimination phase.
- **MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES** 345
Control the maximum number of times the eliminator is tried.
- **MSK_IPAR_PRESOLVE_ELIMINATOR_USE** 345
Controls whether free or implied free variables are eliminated from the problem.

- **MSK_IPAR_PRESOLVE_LEVEL** 346
Currently not used.
- **MSK_IPAR_PRESOLVE_LINDEP_USE** 346
Controls whether the linear constraints are checked for linear dependencies.
- **MSK_IPAR_PRESOLVE_LINDEP_WORK_LIM** 346
Controls linear dependency check in presolve.
- **MSK_DPAR_PRESOLVE_TOL_AIJ** 288
Absolute zero tolerance employed for constraint coefficients in the presolve.
- **MSK_DPAR_PRESOLVE_TOL_LIN_DEP** 289
Controls when a constraint is determined to be linearly dependent.
- **MSK_DPAR_PRESOLVE_TOL_S** 289
Absolute zero tolerance employed for slack variables in the presolve.
- **MSK_DPAR_PRESOLVE_TOL_X** 289
Absolute zero tolerance employed for variables in the presolve.
- **MSK_IPAR_PRESOLVE_USE** 346
Controls whether the presolve is applied to a problem before it is optimized.

E.1.12 Termination criterion parameters.

Parameters which define termination and optimality criteria and related information.

- **MSK_DPAR_BASIS_REL_TOL_S** 271
Maximum relative dual bound violation allowed in an optimal basic solution.
- **MSK_DPAR_BASIS_TOL_S** 271
Maximum absolute dual bound violation in an optimal basic solution.
- **MSK_DPAR_BASIS_TOL_X** 271
Maximum absolute primal bound violation allowed in an optimal basic solution.
- **MSK_IPAR_BI_MAX_ITERATIONS** 308
Maximum number of iterations after basis identification.
- **MSK_DPAR_INTPNT_CO_TOL_DFEAS** 275
Dual feasibility tolerance used by the conic interior-point optimizer.
- **MSK_DPAR_INTPNT_CO_TOL_INFEAS** 276
Infeasibility tolerance for the conic solver.

- **MSK_DPAR_INTPNT_CO_TOL_MU_RED** 276
Optimality tolerance for the conic solver.
- **MSK_DPAR_INTPNT_CO_TOL_NEAR_REL** 276
Optimality tolerance for the conic solver.
- **MSK_DPAR_INTPNT_CO_TOL_PFEAS** 277
Primal feasibility tolerance used by the conic interior-point optimizer.
- **MSK_DPAR_INTPNT_CO_TOL_REL_GAP** 277
Relative gap termination tolerance used by the conic interior-point optimizer.
- **MSK_IPAR_INTPNT_MAX_ITERATIONS** 316
Controls the maximum number of iterations allowed in the interior-point optimizer.
- **MSK_DPAR_INTPNT_NL_TOL_DFEAS** 277
Dual feasibility tolerance used when a nonlinear model is solved.
- **MSK_DPAR_INTPNT_NL_TOL_MU_RED** 278
Relative complementarity gap tolerance.
- **MSK_DPAR_INTPNT_NL_TOL_NEAR_REL** 278
Nonlinear solver optimality tolerance parameter.
- **MSK_DPAR_INTPNT_NL_TOL_PFEAS** 278
Primal feasibility tolerance used when a nonlinear model is solved.
- **MSK_DPAR_INTPNT_NL_TOL_REL_GAP** 279
Relative gap termination tolerance for nonlinear problems.
- **MSK_DPAR_INTPNT_TOL_DFEAS** 279
Dual feasibility tolerance used for linear and quadratic optimization problems.
- **MSK_DPAR_INTPNT_TOL_INFEAS** 280
Nonlinear solver infeasibility tolerance parameter.
- **MSK_DPAR_INTPNT_TOL_MU_RED** 280
Relative complementarity gap tolerance.
- **MSK_DPAR_INTPNT_TOL_PFEAS** 281
Primal feasibility tolerance used for linear and quadratic optimization problems.
- **MSK_DPAR_INTPNT_TOL_REL_GAP** 281
Relative gap termination tolerance.
- **MSK_DPAR_LOWER_OBJ_CUT** 282
Objective bound.

- **MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH** 282
Objective bound.
- **MSK_DPAR_MIO_DISABLE_TERM_TIME** 283
Certain termination criteria is disabled within the mixed-integer optimizer for period time specified by the parameter.
- **MSK_IPAR_MIO_MAX_NUM_BRANCHES** 335
Maximum number of branches allowed during the branch and bound search.
- **MSK_IPAR_MIO_MAX_NUM_SOLUTIONS** 336
Controls how many feasible solutions the mixed-integer optimizer investigates.
- **MSK_DPAR_MIO_MAX_TIME** 284
Time limit for the mixed-integer optimizer.
- **MSK_DPAR_MIO_NEAR_TOL_REL_GAP** 285
The mixed-integer optimizer is terminated when this tolerance is satisfied.
- **MSK_DPAR_MIO_REL_GAP_CONST** 285
This value is used to compute the relative gap for the solution to an integer optimization problem.
- **MSK_DPAR_MIO_TOL_REL_GAP** 287
Relative optimality tolerance employed by the mixed-integer optimizer.
- **MSK_DPAR_OPTIMIZER_MAX_TIME** 288
Solver time limit.
- **MSK_IPAR_SIM_MAX_ITERATIONS** 359
Maximum number of iterations that can be used by a simplex optimizer.
- **MSK_DPAR_UPPER_OBJ_CUT** 290
Objective bound.
- **MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH** 291
Objective bound.

E.1.13 Progress call-back parameters.

- **MSK_DPAR_CALLBACK_FREQ** 272
Controls progress call-back frequency.
- **MSK_IPAR_SOLUTION_CALLBACK** 367
Indicates whether solution call-backs will be performed during the optimization.

E.1.14 Non-convex solver parameters.

- **MSK_IPAR_LOG_NONCONVEX** 326
Controls amount of output printed by the nonconvex optimizer.
- **MSK_IPAR_NONCONVEX_MAX_ITERATIONS** 340
Maximum number of iterations that can be used by the nonconvex optimizer.
- **MSK_DPAR_NONCONVEX_TOL_FEAS** 287
Feasibility tolerance used by the nonconvex optimizer.
- **MSK_DPAR_NONCONVEX_TOL_OPT** 288
Optimality tolerance used by the nonconvex optimizer.

E.1.15 Feasibility repair parameters.

- **MSK_DPAR_FEASREPAIR_TOL** 275
Tolerance for constraint enforcing upper bound on sum of weighted violations in feasibility repair.

E.1.16 Optimization system parameters.

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

- **MSK_IPAR_AUTO_UPDATE_SOL_INFO** 306
Controls whether the solution information items are automatically updated after an optimization is performed.
- **MSK_IPAR_CACHE_LICENSE** 309
Control license caching.
- **MSK_IPAR_CACHE_SIZE_L1** 309
Specifies the size of the level 1 cache of the processor.
- **MSK_IPAR_CACHE_SIZE_L2** 309
Specifies the size of the level 2 cache of the processor.
- **MSK_IPAR_CPU_TYPE** 312
Specifies the CPU type.
- **MSK_IPAR_INTPNT_NUM_THREADS** 317
Controls the number of threads employed by the interior-point optimizer. If set to a positive number MOSEK will use this number of threads. If zero the number of threads used will equal the number of cores detected on the machine.

- **MSK_IPAR_LICENSE_CACHE_TIME** 320
Setting this parameter no longer has any effect.
- **MSK_IPAR_LICENSE_CHECK_TIME** 320
Controls the license manager client behavior.
- **MSK_IPAR_LICENSE_WAIT** 322
Controls if MOSEK should queue for a license if none is available.
- **MSK_IPAR_LOG_STORAGE** 330
Controls the memory related log information.
- **MSK_IPAR_TIMING_LEVEL** 368
Controls the a amount of timing performed inside MOSEK.

E.1.17 Output information parameters.

- **MSK_IPAR_INFEAS_REPORT_LEVEL** 314
Controls the contents of the infeasibility report.
- **MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS** 321
Controls license manager client behavior.
- **MSK_IPAR_LOG** 322
Controls the amount of log information.
- **MSK_IPAR_LOG_BI** 322
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** 323
Controls the logging frequency.
- **MSK_IPAR_LOG_CUT_SECOND_OPT** 324
Controls the reduction in the log levels for the second and any subsequent optimizations.
- **MSK_IPAR_LOG_FACTOR** 324
If turned on, then the factor log lines are added to the log.
- **MSK_IPAR_LOG_FEASREPAIR** 324
Controls the amount of output printed when performing feasibility repair.
- **MSK_IPAR_LOG_FILE** 325
If turned on, then some log info is printed when a file is written or read.

- **MSK_IPAR_LOG_HEAD** 325
If turned on, then a header line is added to the log.
- **MSK_IPAR_LOG_INFEAS_ANA** 325
Controls log level for the infeasibility analyzer.
- **MSK_IPAR_LOG_INTPNT** 326
Controls the amount of log information from the interior-point optimizers.
- **MSK_IPAR_LOG_MIO** 326
Controls the amount of log information from the mixed-integer optimizers.
- **MSK_IPAR_LOG_MIO_FREQ** 326
The mixed-integer solver logging frequency.
- **MSK_IPAR_LOG_NONCONVEX** 326
Controls amount of output printed by the nonconvex optimizer.
- **MSK_IPAR_LOG_OPTIMIZER** 327
Controls the amount of general optimizer information that is logged.
- **MSK_IPAR_LOG_ORDER** 327
If turned on, then factor lines are added to the log.
- **MSK_IPAR_LOG_PARAM** 327
Controls the amount of information printed out about parameter changes.
- **MSK_IPAR_LOG_RESPONSE** 328
Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.
- **MSK_IPAR_LOG_SENSITIVITY** 328
Control logging in sensitivity analyzer.
- **MSK_IPAR_LOG_SENSITIVITY_OPT** 328
Control logging in sensitivity analyzer.
- **MSK_IPAR_LOG_SIM** 329
Controls the amount of log information from the simplex optimizers.
- **MSK_IPAR_LOG_SIM_FREQ** 329
Controls simplex logging frequency.
- **MSK_IPAR_LOG_SIM_MINOR** 329
Currently not in use.

- **MSK_IPAR_LOG_SIM_NETWORK_FREQ**.....330
Controls the network simplex logging frequency.
- **MSK_IPAR_LOG_STORAGE**.....330
Controls the memory related log information.
- **MSK_IPAR_MAX_NUM_WARNINGS**.....331
Waning level. A higher value results in more warnings.
- **MSK_IPAR_WARNING_LEVEL**.....368
Warning level.

E.1.18 Extra information about the optimization problem.

- **MSK_IPAR_OBJECTIVE_SENSE**.....340
If the objective sense for the task is undefined, then the value of this parameter is used as the default objective sense.

E.1.19 Overall solver parameters.

- **MSK_IPAR_BI_CLEAN_OPTIMIZER**.....307
Controls which simplex optimizer is used in the clean-up phase.
- **MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS**.....310
The maximum number of simultaneous optimizations that will be started by the concurrent optimizer.
- **MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX**.....311
Priority of the dual simplex algorithm when selecting solvers for concurrent optimization.
- **MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX**.....311
Priority of the free simplex optimizer when selecting solvers for concurrent optimization.
- **MSK_IPAR_CONCURRENT_PRIORITY_INTPNT**.....311
Priority of the interior-point algorithm when selecting solvers for concurrent optimization.
- **MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX**.....312
Priority of the primal simplex algorithm when selecting solvers for concurrent optimization.
- **MSK_IPAR_DATA_CHECK**.....312
Enable data checking for debug purposes.

- **MSK_IPAR_FEASREPAIR_OPTIMIZE** 313
Controls which type of feasibility analysis is to be performed.
- **MSK_IPAR_INFEAS_PREFER_PRIMAL** 313
Controls which certificate is used if both primal- and dual- certificate of infeasibility is available.
- **MSK_IPAR_LICENSE_WAIT** 322
Controls if MOSEK should queue for a license if none is available.
- **MSK_IPAR_MIO_CONT_SOL** 332
Controls the meaning of interior-point and basic solutions in mixed integer problems.
- **MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER** 335
Controls the size of the local search space when doing local branching.
- **MSK_IPAR_MIO_MODE** 336
Turns on/off the mixed-integer mode.
- **MSK_IPAR_OPTIMIZER** 343
Controls which optimizer is used to optimize the task.
- **MSK_IPAR_PRESOLVE_LEVEL** 346
Currently not used.
- **MSK_IPAR_PRESOLVE_USE** 346
Controls whether the presolve is applied to a problem before it is optimized.
- **MSK_IPAR_SENSITIVITY_OPTIMIZER** 354
Controls which optimizer is used for optimal partition sensitivity analysis.
- **MSK_IPAR_SENSITIVITY_TYPE** 355
Controls which type of sensitivity analysis is to be performed.
- **MSK_IPAR_SOLUTION_CALLBACK** 367
Indicates whether solution call-backs will be performed during the optimization.

E.1.20 Behavior of the optimization task.

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

- **MSK_IPAR_ALLOC_ADD_QNZ** 305
Controls how the quadratic matrixes are extended.
- **MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL** 379
Feasibility repair name violation name.

- **MSK_IPAR_READ_ADD_ANZ** 347
Controls how the constraint matrix is extended.
- **MSK_IPAR_READ_ADD_CON** 347
Additional number of constraints that is made room for in the problem.
- **MSK_IPAR_READ_ADD_CONE** 348
Additional number of conic constraints that is made room for in the problem.
- **MSK_IPAR_READ_ADD_QNZ** 348
Controls how the quadratic matrixes are extended.
- **MSK_IPAR_READ_ADD_VAR** 348
Additional number of variables that is made room for in the problem.
- **MSK_IPAR_READ_ANZ** 348
Controls the expected number of constraint non-zeros.
- **MSK_IPAR_READ_CON** 349
Controls the expected number of constraints.
- **MSK_IPAR_READ_CONE** 349
Controls the expected number of conic constraints.
- **MSK_IPAR_READ_QNZ** 353
Controls the expected number of quadratic non-zeros.
- **MSK_IPAR_READ_TASK_IGNORE_PARAM** 354
Controls what information is used from the task files.
- **MSK_IPAR_READ_VAR** 354
Controls the expected number of variables.
- **MSK_IPAR_WRITE_TASK_INC_SOL** 376
Controls whether the solutions are stored in the task file too.

E.1.21 Data input/output parameters.

Parameters defining the behavior of data readers and writers.

- **MSK_SPAR_BAS_SOL_FILE_NAME** 378
Name of the bas solution file.
- **MSK_SPAR_DATA_FILE_NAME** 378
Data are read and written to this file.

- **MSK_SPAR_DEBUG_FILE_NAME** 378
MOSEK debug file.
- **MSK_IPAR_INFEAS_REPORT_AUTO** 314
Turns the feasibility report on or off.
- **MSK_SPAR_INT_SOL_FILE_NAME** 379
Name of the int solution file.
- **MSK_SPAR_ITR_SOL_FILE_NAME** 379
Name of the itr solution file.
- **MSK_IPAR_LOG_FILE** 325
If turned on, then some log info is printed when a file is written or read.
- **MSK_IPAR_LP_WRITE_IGNORE_INCOMPATIBLE_ITEMS** 330
Controls the result of writing a problem containing incompatible items to an LP file.
- **MSK_IPAR_OPF_MAX_TERMS_PER_LINE** 341
The maximum number of terms (linear and quadratic) per line when an OPF file is written.
- **MSK_IPAR_OPF_WRITE_HEADER** 341
Write a text header with date and MOSEK version in an OPF file.
- **MSK_IPAR_OPF_WRITE_HINTS** 341
Write a hint section with problem dimensions in the beginning of an OPF file.
- **MSK_IPAR_OPF_WRITE_PARAMETERS** 341
Write a parameter section in an OPF file.
- **MSK_IPAR_OPF_WRITE_PROBLEM** 342
Write objective, constraints, bounds etc. to an OPF file.
- **MSK_IPAR_OPF_WRITE_SOL_BAS** 342
Controls what is written to the OPF files.
- **MSK_IPAR_OPF_WRITE_SOL_ITG** 342
Controls what is written to the OPF files.
- **MSK_IPAR_OPF_WRITE_SOL_ITR** 343
Controls what is written to the OPF files.
- **MSK_IPAR_OPF_WRITE_SOLUTIONS** 343
Enable inclusion of solutions in the OPF files.

• MSK_SPAR_PARAM_COMMENT_SIGN	379
Solution file comment character.	
• MSK_IPAR_PARAM_READ_CASE_NAME	344
If turned on, then names in the parameter file are case sensitive.	
• MSK_SPAR_PARAM_READ_FILE_NAME	380
Modifications to the parameter database is read from this file.	
• MSK_IPAR_PARAM_READ_IGN_ERROR	344
If turned on, then errors in paramter settings is ignored.	
• MSK_SPAR_PARAM_WRITE_FILE_NAME	380
The parameter database is written to this file.	
• MSK_IPAR_READ_ADD_ANZ	347
Controls how the constraint matrix is extended.	
• MSK_IPAR_READ_ADD_CON	347
Additional number of constraints that is made room for in the problem.	
• MSK_IPAR_READ_ADD_CONE	348
Additional number of conic constraints that is made room for in the problem.	
• MSK_IPAR_READ_ADD_QNZ	348
Controls how the quadratic matrixes are extended.	
• MSK_IPAR_READ_ADD_VAR	348
Additional number of variables that is made room for in the problem.	
• MSK_IPAR_READ_ANZ	348
Controls the expected number of constraint non-zeros.	
• MSK_IPAR_READ_CON	349
Controls the expected number of constraints.	
• MSK_IPAR_READ_CONE	349
Controls the expected number of conic constraints.	
• MSK_IPAR_READ_DATA_COMPRESSED	349
Controls the input file decompression.	
• MSK_IPAR_READ_DATA_FORMAT	350
Format of the data file to be read.	
• MSK_IPAR_READ_KEEP_FREE_CON	350
Controls whether the free constraints are included in the problem.	

- **MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU** 350
Controls how the LP files are interpreted.
- **MSK_IPAR_READ_LP_QUOTED_NAMES** 351
If a name is in quotes when reading an LP file, the quotes will be removed.
- **MSK_SPAR_READ_MPS_BOU_NAME** 380
Name of the BOUNDS vector used. An empty name means that the first BOUNDS vector is used.
- **MSK_IPAR_READ_MPS_FORMAT** 351
Controls how strictly the MPS file reader interprets the MPS format.
- **MSK_IPAR_READ_MPS_KEEP_INT** 352
Controls if integer constraints are read.
- **MSK_SPAR_READ_MPS_OBJ_NAME** 380
Objective name in the MPS file.
- **MSK_IPAR_READ_MPS_OBJ_SENSE** 352
Controls the MPS format extensions.
- **MSK_IPAR_READ_MPS_QUOTED_NAMES** 352
Controls the MPS format extensions.
- **MSK_SPAR_READ_MPS_RAN_NAME** 381
Name of the RANGE vector used. An empty name means that the first RANGE vector is used.
- **MSK_IPAR_READ_MPS_RELAX** 352
Controls the meaning of integer constraints.
- **MSK_SPAR_READ_MPS_RHS_NAME** 381
Name of the RHS used. An empty name means that the first RHS vector is used.
- **MSK_IPAR_READ_MPS_WIDTH** 353
Controls the maximal number of characters allowed in one line of the MPS file.
- **MSK_IPAR_READ_Q_MODE** 353
Controls how the Q matrices are read from the MPS file.
- **MSK_IPAR_READ_QNZ** 353
Controls the expected number of quadratic non-zeros.
- **MSK_IPAR_READ_TASK_IGNORE_PARAM** 354
Controls what information is used from the task files.

• MSK_IPAR_READ_VAR	354
Controls the expected number of variables.	
• MSK_SPAR_SOL_FILTER_XC_LOW	381
Solution file filter.	
• MSK_SPAR_SOL_FILTER_XC_UPR	382
Solution file filter.	
• MSK_SPAR_SOL_FILTER_XX_LOW	382
Solution file filter.	
• MSK_SPAR_SOL_FILTER_XX_UPR	382
Solution file filter.	
• MSK_IPAR_SOL_QUOTED_NAMES	366
Controls the solution file format.	
• MSK_IPAR_SOL_READ_NAME_WIDTH	367
Controls the input solution file format.	
• MSK_IPAR_SOL_READ_WIDTH	367
Controls the input solution file format.	
• MSK_SPAR_STAT_FILE_NAME	383
Statistics file name.	
• MSK_SPAR_STAT_KEY	383
Key used when writing the summary file.	
• MSK_SPAR_STAT_NAME	383
Name used when writing the statistics file.	
• MSK_IPAR_WRITE_BAS_CONSTRAINTS	368
Controls the basic solution file format.	
• MSK_IPAR_WRITE_BAS_HEAD	368
Controls the basic solution file format.	
• MSK_IPAR_WRITE_BAS_VARIABLES	369
Controls the basic solution file format.	
• MSK_IPAR_WRITE_DATA_COMPRESSED	369
Controls output file compression.	
• MSK_IPAR_WRITE_DATA_FORMAT	369
Controls the output file format.	

- **MSK_IPAR_WRITE_DATA_PARAM** 370
Controls output file data.
- **MSK_IPAR_WRITE_FREE_CON** 370
Controls the output file data.
- **MSK_IPAR_WRITE_GENERIC_NAMES** 370
Controls the output file data.
- **MSK_IPAR_WRITE_GENERIC_NAMES_IO** 371
Index origin used in generic names.
- **MSK_IPAR_WRITE_INT_CONSTRAINTS** 371
Controls the integer solution file format.
- **MSK_IPAR_WRITE_INT_HEAD** 371
Controls the integer solution file format.
- **MSK_IPAR_WRITE_INT_VARIABLES** 372
Controls the integer solution file format.
- **MSK_SPAR_WRITE_LP_GEN_VAR_NAME** 383
Added variable names in the LP files.
- **MSK_IPAR_WRITE_LP_LINE_WIDTH** 372
Controls the LP output file format.
- **MSK_IPAR_WRITE_LP_QUOTED_NAMES** 372
Controls LP output file format.
- **MSK_IPAR_WRITE_LP_STRICT_FORMAT** 373
Controls whether LP output files satisfy the LP format strictly.
- **MSK_IPAR_WRITE_LP_TERMS_PER_LINE** 373
Controls the LP output file format.
- **MSK_IPAR_WRITE_MPS_INT** 373
Controls the output file data.
- **MSK_IPAR_WRITE_MPS_OBJ_SENSE** 373
Controls the output file data.
- **MSK_IPAR_WRITE_MPS_QUOTED_NAMES** 374
Controls the output file data.
- **MSK_IPAR_WRITE_MPS_STRICT** 374
Controls the output MPS file format.

- **MSK_IPAR_WRITE_PRECISION** 374
Controls data precision employed in when writing an MPS file.
- **MSK_IPAR_WRITE_SOL_CONSTRAINTS** 375
Controls the solution file format.
- **MSK_IPAR_WRITE_SOL_HEAD** 375
Controls solution file format.
- **MSK_IPAR_WRITE_SOL_VARIABLES** 375
Controls the solution file format.
- **MSK_IPAR_WRITE_TASK_INC_SOL** 376
Controls whether the solutions are stored in the task file too.
- **MSK_IPAR_WRITE_XML_MODE** 376
Controls if linear coefficients should be written by row or column when writing in the XML file format.

E.1.22 Analysis parameters.

Parameters controlling the behaviour of the problem and solution analyzers.

- **MSK_IPAR_ANA_SOL_BASIS** 305
Controls whether the basis matrix is analyzed in solution analyzer.
- **MSK_DPAR_ANA_SOL_INFEAS_TOL** 270
If a constraint violates its bound with an amount larger than this value, the constraint name, index and violation will be printed by the solution analyzer.
- **MSK_IPAR_ANA_SOL_PRINT_VIOLATED** 306
Controls whether a list of violated constraints is printed.

E.1.23 Solution input/output parameters.

Parameters defining the behavior of solution reader and writer.

- **MSK_SPAR_BAS_SOL_FILE_NAME** 378
Name of the bas solution file.
- **MSK_IPAR_INFEAS_REPORT_AUTO** 314
Turns the feasibility report on or off.
- **MSK_SPAR_INT_SOL_FILE_NAME** 379
Name of the int solution file.

- **MSK_SPAR_ITR_SOL_FILE_NAME** 379
Name of the itr solution file.
- **MSK_IPAR_SOL_FILTER_KEEP_BASIC** 366
Controls the license manager client behavior.
- **MSK_IPAR_SOL_FILTER_KEEP_RANGED** 366
Control the contents of the solution files.
- **MSK_SPAR_SOL_FILTER_XC_LOW** 381
Solution file filter.
- **MSK_SPAR_SOL_FILTER_XC_UPR** 382
Solution file filter.
- **MSK_SPAR_SOL_FILTER_XX_LOW** 382
Solution file filter.
- **MSK_SPAR_SOL_FILTER_XX_UPR** 382
Solution file filter.
- **MSK_IPAR_SOL_QUOTED_NAMES** 366
Controls the solution file format.
- **MSK_IPAR_SOL_READ_NAME_WIDTH** 367
Controls the input solution file format.
- **MSK_IPAR_SOL_READ_WIDTH** 367
Controls the input solution file format.
- **MSK_IPAR_WRITE_BAS_CONSTRAINTS** 368
Controls the basic solution file format.
- **MSK_IPAR_WRITE_BAS_HEAD** 368
Controls the basic solution file format.
- **MSK_IPAR_WRITE_BAS_VARIABLES** 369
Controls the basic solution file format.
- **MSK_IPAR_WRITE_INT_CONSTRAINTS** 371
Controls the integer solution file format.
- **MSK_IPAR_WRITE_INT_HEAD** 371
Controls the integer solution file format.
- **MSK_IPAR_WRITE_INT_VARIABLES** 372
Controls the integer solution file format.

- **MSK_IPAR_WRITE_SOL_CONSTRAINTS** 375
Controls the solution file format.
- **MSK_IPAR_WRITE_SOL_HEAD** 375
Controls solution file format.
- **MSK_IPAR_WRITE_SOL_VARIABLES** 375
Controls the solution file format.

E.1.24 Infeasibility report parameters.

- **MSK_IPAR_INFEAS_GENERIC_NAMES** 313
Controls the contents of the infeasibility report.
- **MSK_IPAR_INFEAS_REPORT_LEVEL** 314
Controls the contents of the infeasibility report.
- **MSK_IPAR_LOG_INFEAS_ANA** 325
Controls log level for the infeasibility analyzer.

E.1.25 License manager parameters.

- **MSK_IPAR_LICENSE_ALLOW_OVERUSE** 320
Controls if license overuse is allowed when caching licenses
- **MSK_IPAR_LICENSE_CACHE_TIME** 320
Setting this parameter no longer has any effect.
- **MSK_IPAR_LICENSE_CHECK_TIME** 320
Controls the license manager client behavior.
- **MSK_IPAR_LICENSE_DEBUG** 321
Controls the license manager client debugging behavior.
- **MSK_IPAR_LICENSE_PAUSE_TIME** 321
Controls license manager client behavior.
- **MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS** 321
Controls license manager client behavior.
- **MSK_IPAR_LICENSE_WAIT** 322
Controls if MOSEK should queue for a license if none is available.

E.1.26 Data check parameters.

These parameters defines data checking settings and problem data tolerances, i.e. which values are rounded to 0 or infinity, and which values are large or small enough to produce a warning.

- **MSK_IPAR_CHECK_CONVEXITY** 310
Specify the level of convexity check on quadratic problems
- **MSK_IPAR_CHECK_TASK_DATA** 310
If this feature is turned on, then the task data is checked for bad values i.e. NaNs. before an optimization is performed.
- **MSK_DPAR_DATA_TOL_AIJ** 272
Data tolerance threshold.
- **MSK_DPAR_DATA_TOL_AIJ_HUGE** 273
Data tolerance threshold.
- **MSK_DPAR_DATA_TOL_AIJ_LARGE** 273
Data tolerance threshold.
- **MSK_DPAR_DATA_TOL_BOUND_INF** 273
Data tolerance threshold.
- **MSK_DPAR_DATA_TOL_BOUND_WRN** 273
Data tolerance threshold.
- **MSK_DPAR_DATA_TOL_C_HUGE** 274
Data tolerance threshold.
- **MSK_DPAR_DATA_TOL_CJ_LARGE** 274
Data tolerance threshold.
- **MSK_DPAR_DATA_TOL_QIJ** 274
Data tolerance threshold.
- **MSK_DPAR_DATA_TOL_X** 275
Data tolerance threshold.
- **MSK_IPAR_LOG_CHECK_CONVEXITY** 323
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.

E.1.27 Debugging parameters.

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

- **MSK_IPAR_AUTO_SORT_A_BEFORE_OPT** 306
Controls whether the elements in each column of A are sorted before an optimization is performed.
- **MSK_IPAR_CHECK_TASK_DATA** 310
If this feature is turned on, then the task data is checked for bad values i.e. NaNs. before an optimization is performed.

E.2 Double parameters

- **MSK_DPAR_ANA_SOL_INFEAS_TOL** 270
If a constraint violates its bound with an amount larger than this value, the constraint name, index and violation will be printed by the solution analyzer.
- **MSK_DPAR_BASIS_REL_TOL_S** 271
Maximum relative dual bound violation allowed in an optimal basic solution.
- **MSK_DPAR_BASIS_TOL_S** 271
Maximum absolute dual bound violation in an optimal basic solution.
- **MSK_DPAR_BASIS_TOL_X** 271
Maximum absolute primal bound violation allowed in an optimal basic solution.
- **MSK_DPAR_CALLBACK_FREQ** 272
Controls progress call-back frequency.
- **MSK_DPAR_CHECK_CONVEXITY_REL_TOL** 272
Convexity check tolerance.
- **MSK_DPAR_DATA_TOL_AIJ** 272
Data tolerance threshold.
- **MSK_DPAR_DATA_TOL_AIJ_HUGE** 273
Data tolerance threshold.
- **MSK_DPAR_DATA_TOL_AIJ_LARGE** 273
Data tolerance threshold.
- **MSK_DPAR_DATA_TOL_BOUND_INF** 273
Data tolerance threshold.

- **MSK_DPAR_DATA_TOL_BOUND_WRN** 273
Data tolerance threshold.
- **MSK_DPAR_DATA_TOL_C_HUGE** 274
Data tolerance threshold.
- **MSK_DPAR_DATA_TOL_CJ_LARGE** 274
Data tolerance threshold.
- **MSK_DPAR_DATA_TOL_QIJ** 274
Data tolerance threshold.
- **MSK_DPAR_DATA_TOL_X** 275
Data tolerance threshold.
- **MSK_DPAR_FEASREPAIR_TOL** 275
Tolerance for constraint enforcing upper bound on sum of weighted violations in feasibility repair.
- **MSK_DPAR_INTPNT_CO_TOL_DFEAS** 275
Dual feasibility tolerance used by the conic interior-point optimizer.
- **MSK_DPAR_INTPNT_CO_TOL_INFfeas** 276
Infeasibility tolerance for the conic solver.
- **MSK_DPAR_INTPNT_CO_TOL_MU_RED** 276
Optimality tolerance for the conic solver.
- **MSK_DPAR_INTPNT_CO_TOL_NEAR_REL** 276
Optimality tolerance for the conic solver.
- **MSK_DPAR_INTPNT_CO_TOL_PFEAS** 277
Primal feasibility tolerance used by the conic interior-point optimizer.
- **MSK_DPAR_INTPNT_CO_TOL_REL_GAP** 277
Relative gap termination tolerance used by the conic interior-point optimizer.
- **MSK_DPAR_INTPNT_NL_MERIT_BAL** 277
Controls if the complementarity and infeasibility is converging to zero at about equal rates.
- **MSK_DPAR_INTPNT_NL_TOL_DFEAS** 277
Dual feasibility tolerance used when a nonlinear model is solved.
- **MSK_DPAR_INTPNT_NL_TOL_MU_RED** 278
Relative complementarity gap tolerance.

- **MSK_DPAR_INTPNT_NL_TOL_NEAR_REL** 278
Nonlinear solver optimality tolerance parameter.
- **MSK_DPAR_INTPNT_NL_TOL_PFEAS** 278
Primal feasibility tolerance used when a nonlinear model is solved.
- **MSK_DPAR_INTPNT_NL_TOL_REL_GAP** 279
Relative gap termination tolerance for nonlinear problems.
- **MSK_DPAR_INTPNT_NL_TOL_REL_STEP** 279
Relative step size to the boundary for general nonlinear optimization problems.
- **MSK_DPAR_INTPNT_TOL_DFEAS** 279
Dual feasibility tolerance used for linear and quadratic optimization problems.
- **MSK_DPAR_INTPNT_TOL_DSAFE** 279
Controls the interior-point dual starting point.
- **MSK_DPAR_INTPNT_TOL_INFEAS** 280
Nonlinear solver infeasibility tolerance parameter.
- **MSK_DPAR_INTPNT_TOL_MU_RED** 280
Relative complementarity gap tolerance.
- **MSK_DPAR_INTPNT_TOL_PATH** 280
interior-point centering aggressiveness.
- **MSK_DPAR_INTPNT_TOL_PFEAS** 281
Primal feasibility tolerance used for linear and quadratic optimization problems.
- **MSK_DPAR_INTPNT_TOL_PSAFE** 281
Controls the interior-point primal starting point.
- **MSK_DPAR_INTPNT_TOL_REL_GAP** 281
Relative gap termination tolerance.
- **MSK_DPAR_INTPNT_TOL_REL_STEP** 281
Relative step size to the boundary for linear and quadratic optimization problems.
- **MSK_DPAR_INTPNT_TOL_STEP_SIZE** 282
If the step size falls below the value of this parameter, then the interior-point optimizer assumes that it is stalled. It it does not not make any progress.
- **MSK_DPAR_LOWER_OBJ_CUT** 282
Objective bound.

- **MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH** 282
Objective bound.
- **MSK_DPAR_MIO_DISABLE_TERM_TIME** 283
Certain termination criteria is disabled within the mixed-integer optimizer for period time specified by the parameter.
- **MSK_DPAR_MIO_HEURISTIC_TIME** 283
Time limit for the mixed-integer heuristics.
- **MSK_DPAR_MIO_MAX_TIME** 284
Time limit for the mixed-integer optimizer.
- **MSK_DPAR_MIO_MAX_TIME_APRX_OPT** 284
Time limit for the mixed-integer optimizer.
- **MSK_DPAR_MIO_NEAR_TOL_ABS_GAP** 284
Relaxed absolute optimality tolerance employed by the mixed-integer optimizer.
- **MSK_DPAR_MIO_NEAR_TOL_REL_GAP** 285
The mixed-integer optimizer is terminated when this tolerance is satisfied.
- **MSK_DPAR_MIO_REL_ADD_CUT_LIMITED** 285
Controls cut generation for mixed-integer optimizer.
- **MSK_DPAR_MIO_REL_GAP_CONST** 285
This value is used to compute the relative gap for the solution to an integer optimization problem.
- **MSK_DPAR_MIO_TOL_ABS_GAP** 286
Absolute optimality tolerance employed by the mixed-integer optimizer.
- **MSK_DPAR_MIO_TOL_ABS_RELAX_INT** 286
Integer constraint tolerance.
- **MSK_DPAR_MIO_TOL_FEAS** 286
Feasibility tolerance for mixed integer solver. Any solution with maximum infeasibility below this value will be considered feasible.
- **MSK_DPAR_MIO_TOL_REL_GAP** 287
Relative optimality tolerance employed by the mixed-integer optimizer.
- **MSK_DPAR_MIO_TOL_REL_RELAX_INT** 287
Integer constraint tolerance.

- **MSK_DPAR_MIO_TOL_X** 287
Absolute solution tolerance used in mixed-integer optimizer.
- **MSK_DPAR_NONCONVEX_TOL_FEAS** 287
Feasibility tolerance used by the nonconvex optimizer.
- **MSK_DPAR_NONCONVEX_TOL_OPT** 288
Optimality tolerance used by the nonconvex optimizer.
- **MSK_DPAR_OPTIMIZER_MAX_TIME** 288
Solver time limit.
- **MSK_DPAR_PRESOLVE_TOL_AIJ** 288
Absolute zero tolerance employed for constraint coefficients in the presolve.
- **MSK_DPAR_PRESOLVE_TOL_LIN_DEP** 289
Controls when a constraint is determined to be linearly dependent.
- **MSK_DPAR_PRESOLVE_TOL_S** 289
Absolute zero tolerance employed for slack variables in the presolve.
- **MSK_DPAR_PRESOLVE_TOL_X** 289
Absolute zero tolerance employed for variables in the presolve.
- **MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL** 289
This parameter determines when columns are dropped in incomplete cholesky factorization doing reformulation of quadratic problems.
- **MSK_DPAR_SIM_LU_TOL_REL_PIV** 290
Relative pivot tolerance employed when computing the LU factorization of the basis matrix.
- **MSK_DPAR_SIMPLEX_ABS_TOL_PIV** 290
Absolute pivot tolerance employed by the simplex optimizers.
- **MSK_DPAR_UPPER_OBJ_CUT** 290
Objective bound.
- **MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH** 291
Objective bound.
- **ana_sol_infeas_tol**

Corresponding constant:

MSK_DPAR_ANA_SOL_INFEAS_TOL

Description:

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

Possible Values:

Any number between 0.0 and +inf.

Default value:

+1e-8

- `basis_rel_tol_s`

Corresponding constant:

MSK_DPAR_BASIS_REL_TOL_S

Description:

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

Possible Values:

Any number between 0.0 and +inf.

Default value:

1.0e-12

- `basis_tol_s`

Corresponding constant:

MSK_DPAR_BASIS_TOL_S

Description:

Maximum absolute dual bound violation in an optimal basic solution.

Possible Values:

Any number between 1.0e-9 and +inf.

Default value:

1.0e-6

- `basis_tol_x`

Corresponding constant:

MSK_DPAR_BASIS_TOL_X

Description:

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

Possible Values:

Any number between 1.0e-9 and +inf.

Default value:

1.0e-6

- `callback_freq`

Corresponding constant:

MSK_DPAR_CALLBACK_FREQ

Description:

Controls the time between calls to the progress call-back function. Hence, if the value of this parameter is for example 10, then the call-back is called approximately each 10 seconds. A negative value is equivalent to infinity.

In general frequent call-backs may hurt the performance.

Possible Values:

Any number between -inf and +inf.

Default value:

-1.0

- `check_convexity_rel_tol`

Corresponding constant:

MSK_DPAR_CHECK_CONVEXITY_REL_TOL

Description:

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

A problem is declared non-convex if negative (positive) pivot elements are detected in the cholesky factor of a matrix which is required to be PSD (NSD). This parameter controls 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:

$$d_i \leq -|Q_{ii}| * \text{check_convexity_rel_tol}$$

Possible Values:

Any number between 0 and +inf.

Default value:

1e-10

- `data_tol_ajj`

Corresponding constant:

MSK_DPAR_DATA_TOL_AIJ

Description:

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

Possible Values:

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

Default value:

1.0e-12

- `data_tol_aij_huge`

Corresponding constant:

MSK_DPAR_DATA_TOL_AIJ_HUGE

Description:

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

Possible Values:

Any number between 0.0 and $+\text{inf}$.

Default value:

1.0e20

- `data_tol_aij_large`

Corresponding constant:

MSK_DPAR_DATA_TOL_AIJ_LARGE

Description:

An element in A which is larger than this value in absolute size causes a warning message to be printed.

Possible Values:

Any number between 0.0 and $+\text{inf}$.

Default value:

1.0e10

- `data_tol_bound_inf`

Corresponding constant:

MSK_DPAR_DATA_TOL_BOUND_INF

Description:

Any bound which in absolute value is greater than this parameter is considered infinite.

Possible Values:

Any number between 0.0 and $+\text{inf}$.

Default value:

1.0e16

- `data_tol_bound_wrn`

Corresponding constant:

MSK_DPAR_DATA_TOL_BOUND_WRN

Description:

If a bound value is larger than this value in absolute size, then a warning message is issued.

Possible Values:

Any number between 0.0 and +inf.

Default value:

1.0e8

- data_tol_c_huge

Corresponding constant:

MSK_DPAR_DATA_TOL_C_HUGE

Description:

An element in c which is larger than the value of this parameter in absolute terms is considered to be huge and generates an error.

Possible Values:

Any number between 0.0 and +inf.

Default value:

1.0e16

- data_tol_cj_large

Corresponding constant:

MSK_DPAR_DATA_TOL_CJ_LARGE

Description:

An element in c which is larger than this value in absolute terms causes a warning message to be printed.

Possible Values:

Any number between 0.0 and +inf.

Default value:

1.0e8

- data_tol_qij

Corresponding constant:

MSK_DPAR_DATA_TOL_QIJ

Description:

Absolute zero tolerance for elements in Q matrices.

Possible Values:

Any number between 0.0 and +inf.

Default value:

1.0e-16

- `data_tol_x`

Corresponding constant:

MSK_DPAR_DATA_TOL_X

Description:

Zero tolerance for constraints and variables i.e. if the distance between the lower and upper bound is less than this value, then the lower and lower bound is considered identical.

Possible Values:

Any number between 0.0 and +inf.

Default value:

1.0e-8

- `feasrepair_tol`

Corresponding constant:

MSK_DPAR_FEASREPAIR_TOL

Description:

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

Possible Values:

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

Default value:

1.0e-10

- `intpnt_co_tol_dfeas`

Corresponding constant:

MSK_DPAR_INTPNT_CO_TOL_DFEAS

Description:

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

Possible Values:

Any number between 0.0 and 1.0.

Default value:

1.0e-8

See also:

MSK_DPAR_INTPNT_CO_TOL_NEAR_REL Optimality tolerance for the conic solver.

- `intpnt_co_tol_infeas`

Corresponding constant:

`MSK_DPAR_INTPNT_CO_TOL_INFEAS`

Description:

Controls when the conic interior-point optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

Possible Values:

Any number between 0.0 and 1.0.

Default value:

`1.0e-8`

- `intpnt_co_tol_mu_red`

Corresponding constant:

`MSK_DPAR_INTPNT_CO_TOL_MU_RED`

Description:

Relative complementarity gap tolerance feasibility tolerance used by the conic interior-point optimizer.

Possible Values:

Any number between 0.0 and 1.0.

Default value:

`1.0e-8`

- `intpnt_co_tol_near_rel`

Corresponding constant:

`MSK_DPAR_INTPNT_CO_TOL_NEAR_REL`

Description:

If MOSEK cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.

Possible Values:

Any number between 1.0 and $+\infty$.

Default value:

`100`

- `intpnt_co_tol_pfeas`

Corresponding constant:

`MSK_DPAR_INTPNT_CO_TOL_PFEAS`

Description:

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

Possible Values:

Any number between 0.0 and 1.0.

Default value:

`1.0e-8`

See also:

`MSK_DPAR_INTPNT_CO_TOL_NEAR_REL` Optimality tolerance for the conic solver.

- `intpnt_co_tol_rel_gap`

Corresponding constant:

`MSK_DPAR_INTPNT_CO_TOL_REL_GAP`

Description:

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

Possible Values:

Any number between 0.0 and 1.0.

Default value:

`1.0e-8`

See also:

`MSK_DPAR_INTPNT_CO_TOL_NEAR_REL` Optimality tolerance for the conic solver.

- `intpnt_nl_merit_bal`

Corresponding constant:

`MSK_DPAR_INTPNT_NL_MERIT_BAL`

Description:

Controls if the complementarity and infeasibility is converging to zero at about equal rates.

Possible Values:

Any number between 0.0 and 0.99.

Default value:

`1.0e-4`

- `intpnt_nl_tol_dfeas`

Corresponding constant:

MSK_DPAR_INTPNT_NL_TOL_DFEAS

Description:

Dual feasibility tolerance used when a nonlinear model is solved.

Possible Values:

Any number between 0.0 and 1.0.

Default value:

1.0e-8

• `intpnt_nl_tol_mu_red`**Corresponding constant:**

MSK_DPAR_INTPNT_NL_TOL_MU_RED

Description:

Relative complementarity gap tolerance.

Possible Values:

Any number between 0.0 and 1.0.

Default value:

1.0e-12

• `intpnt_nl_tol_near_rel`**Corresponding constant:**

MSK_DPAR_INTPNT_NL_TOL_NEAR_REL

Description:

If the MOSEK nonlinear interior-point optimizer cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.

Possible Values:

Any number between 1.0 and +inf.

Default value:

1000.0

• `intpnt_nl_tol_pfeas`**Corresponding constant:**

MSK_DPAR_INTPNT_NL_TOL_PFEAS

Description:

Primal feasibility tolerance used when a nonlinear model is solved.

Possible Values:

Any number between 0.0 and 1.0.

Default value:

1.0e-8

- `intpnt_n1_tol_rel_gap`

Corresponding constant:

MSK_DPAR_INTPNT_NL_TOL_REL_GAP

Description:

Relative gap termination tolerance for nonlinear problems.

Possible Values:

Any number between 1.0e-14 and +inf.

Default value:

1.0e-6

- `intpnt_n1_tol_rel_step`

Corresponding constant:

MSK_DPAR_INTPNT_NL_TOL_REL_STEP

Description:

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

Possible Values:

Any number between 1.0e-4 and 0.9999999.

Default value:

0.995

- `intpnt_tol_dfeas`

Corresponding constant:

MSK_DPAR_INTPNT_TOL_DFEAS

Description:

Dual feasibility tolerance used for linear and quadratic optimization problems.

Possible Values:

Any number between 0.0 and 1.0.

Default value:

1.0e-8

- `intpnt_tol_dsafe`

Corresponding constant:

MSK_DPAR_INTPNT_TOL_DSAFE

Description:

Controls the initial dual starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly.

Possible Values:

Any number between 1.0e-4 and +inf.

Default value:

1.0

- `intpnt_tol_infeas`

Corresponding constant:

MSK_DPAR_INTPNT_TOL_INFEAS

Description:

Controls when the optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

Possible Values:

Any number between 0.0 and 1.0.

Default value:

1.0e-8

- `intpnt_tol_mu_red`

Corresponding constant:

MSK_DPAR_INTPNT_TOL_MU_RED

Description:

Relative complementarity gap tolerance.

Possible Values:

Any number between 0.0 and 1.0.

Default value:

1.0e-16

- `intpnt_tol_path`

Corresponding constant:

MSK_DPAR_INTPNT_TOL_PATH

Description:

Controls how close the interior-point optimizer follows the central path. A large value of this parameter means the central is followed very closely. On numerical unstable problems it may be worthwhile to increase this parameter.

Possible Values:

Any number between 0.0 and 0.9999.

Default value:

1.0e-8

- `intpnt_tol_pfeas`

Corresponding constant:

MSK_DPAR_INTPNT_TOL_PFEAS

Description:

Primal feasibility tolerance used for linear and quadratic optimization problems.

Possible Values:

Any number between 0.0 and 1.0.

Default value:

1.0e-8

- `intpnt_tol_psafe`

Corresponding constant:

MSK_DPAR_INTPNT_TOL_PSAFE

Description:

Controls the initial primal starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly and/or the constraint or variable bounds are very large, then it may be worthwhile to increase this value.

Possible Values:

Any number between 1.0e-4 and +inf.

Default value:

1.0

- `intpnt_tol_rel_gap`

Corresponding constant:

MSK_DPAR_INTPNT_TOL_REL_GAP

Description:

Relative gap termination tolerance.

Possible Values:

Any number between 1.0e-14 and +inf.

Default value:

1.0e-8

- `intpnt_tol_rel_step`

Corresponding constant:

MSK_DPAR_INTPNT_TOL_REL_STEP

Description:

Relative step size to the boundary for linear and quadratic optimization problems.

Possible Values:

Any number between 1.0e-4 and 0.999999.

Default value:

0.9999

- `intpnt_tol_step_size`

Corresponding constant:

MSK_DPAR_INTPNT_TOL_STEP_SIZE

Description:

If the step size falls below the value of this parameter, then the interior-point optimizer assumes that it is stalled. If it does not make any progress.

Possible Values:

Any number between 0.0 and 1.0.

Default value:

1.0e-10

- `lower_obj_cut`

Corresponding constant:

MSK_DPAR_LOWER_OBJ_CUT

Description:If either a primal or dual feasible solution is found proving that the optimal objective value is outside, the interval `[MSK_DPAR_LOWER_OBJ_CUT, MSK_DPAR_UPPER_OBJ_CUT]`, then MOSEK is terminated.**Possible Values:**

Any number between -inf and +inf.

Default value:

-1.0e30

See also:`MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH` Objective bound.

- `lower_obj_cut_finite_trh`

Corresponding constant:

MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH

Description:

If the lower objective cut is less than the value of this parameter value, then the lower objective cut i.e. `MSK_DPAR_LOWER_OBJ_CUT` is treated as $-\infty$.

Possible Values:

Any number between $-\infty$ and $+\infty$.

Default value:

`-0.5e30`

- `mio_disable_term_time`

Corresponding constant:

`MSK_DPAR_MIO_DISABLE_TERM_TIME`

Description:

The termination criteria governed by

- `MSK_IPAR_MIO_MAX_NUM_RELAXS`
- `MSK_IPAR_MIO_MAX_NUM_BRANCHES`
- `MSK_DPAR_MIO_NEAR_TOL_ABS_GAP`
- `MSK_DPAR_MIO_NEAR_TOL_REL_GAP`

is disabled the first n seconds. This parameter specifies the number n . A negative value is identical to infinity i.e. the termination criteria are never checked.

Possible Values:

Any number between $-\infty$ and $+\infty$.

Default value:

`-1.0`

See also:

`MSK_IPAR_MIO_MAX_NUM_RELAXS` Maximum number of relaxations in branch and bound search.

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

`MSK_DPAR_MIO_NEAR_TOL_ABS_GAP` Relaxed absolute optimality tolerance employed by the mixed-integer optimizer.

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

- `mio_heuristic_time`

Corresponding constant:

`MSK_DPAR_MIO_HEURISTIC_TIME`

Description:

Minimum amount of time to be used in the heuristic search for a good feasible integer solution. A negative values implies that the optimizer decides the amount of time to be spent in the heuristic.

Possible Values:

Any number between -inf and +inf.

Default value:

-1.0

- `mio_max_time`

Corresponding constant:

`MSK_DPAR_MIO_MAX_TIME`

Description:

This parameter limits the maximum time spent by the mixed-integer optimizer. A negative number means infinity.

Possible Values:

Any number between -inf and +inf.

Default value:

-1.0

- `mio_max_time_aprx_opt`

Corresponding constant:

`MSK_DPAR_MIO_MAX_TIME_APRX_OPT`

Description:

Number of seconds spent by the mixed-integer optimizer before the `MSK_DPAR_MIO_TOL_REL_RELAX_INT` is applied.

Possible Values:

Any number between 0.0 and +inf.

Default value:

60

- `mio_near_tol_abs_gap`

Corresponding constant:

`MSK_DPAR_MIO_NEAR_TOL_ABS_GAP`

Description:

Relaxed absolute optimality tolerance employed by the mixed-integer optimizer. This termination criteria is delayed. See `MSK_DPAR_MIO_DISABLE_TERM_TIME` for details.

Possible Values:

Any number between 0.0 and +inf.

Default value:

0.0

See also:

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

- `mio_near_tol_rel_gap`

Corresponding constant:

MSK_DPAR_MIO_NEAR_TOL_REL_GAP

Description:

The mixed-integer optimizer is terminated when this tolerance is satisfied. This termination criteria is delayed. See **MSK_DPAR_MIO_DISABLE_TERM_TIME** for details.

Possible Values:

Any number between 0.0 and +inf.

Default value:

1.0e-3

See also:

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

- `mio_rel_add_cut_limited`

Corresponding constant:

MSK_DPAR_MIO_REL_ADD_CUT_LIMITED

Description:

Controls how many cuts the mixed-integer optimizer is allowed to add to the problem. Let α be the value of this parameter and m the number constraints, then mixed-integer optimizer is allowed to αm cuts.

Possible Values:

Any number between 0.0 and 2.0.

Default value:

0.75

- `mio_rel_gap_const`

Corresponding constant:

MSK_DPAR_MIO_REL_GAP_CONST

Description:

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

Possible Values:

Any number between 1.0e-15 and +inf.

Default value:

1.0e-10

- `mio_tol_abs_gap`

Corresponding constant:

MSK_DPAR_MIO_TOL_ABS_GAP

Description:

Absolute optimality tolerance employed by the mixed-integer optimizer.

Possible Values:

Any number between 0.0 and +inf.

Default value:

0.0

- `mio_tol_abs_relax_int`

Corresponding constant:

MSK_DPAR_MIO_TOL_ABS_RELAX_INT

Description:

Absolute relaxation tolerance of the integer constraints. I.e. $\min(|x| - \lfloor x \rfloor, \lceil x \rceil - |x|)$ is less than the tolerance then the integer restrictions assumed to be satisfied.

Possible Values:

Any number between 0.0 and +inf.

Default value:

1.0e-5

- `mio_tol_feas`

Corresponding constant:

MSK_DPAR_MIO_TOL_FEAS

Description:

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

Possible Values:

Any number between 0.0 and +inf.

Default value:

1.0e-7

• `mio_tol_rel_gap`**Corresponding constant:**

MSK_DPAR_MIO_TOL_REL_GAP

Description:

Relative optimality tolerance employed by the mixed-integer optimizer.

Possible Values:

Any number between 0.0 and +inf.

Default value:

1.0e-4

• `mio_tol_rel_relax_int`**Corresponding constant:**

MSK_DPAR_MIO_TOL_REL_RELAX_INT

Description:

Relative relaxation tolerance of the integer constraints. I.e $(\min(|x| - \lfloor x \rfloor, \lceil x \rceil - |x|))$ is less than the tolerance times $|x|$ then the integer restrictions assumed to be satisfied.

Possible Values:

Any number between 0.0 and +inf.

Default value:

1.0e-6

• `mio_tol_x`**Corresponding constant:**

MSK_DPAR_MIO_TOL_X

Description:

Absolute solution tolerance used in mixed-integer optimizer.

Possible Values:

Any number between 0.0 and +inf.

Default value:

1.0e-6

• `nonconvex_tol_feas`**Corresponding constant:**

MSK_DPAR_NONCONVEX_TOL_FEAS

Description:

Feasibility tolerance used by the nonconvex optimizer.

Possible Values:

Any number between 0.0 and +inf.

Default value:

1.0e-6

- `nonconvex_tol_opt`

Corresponding constant:

MSK_DPAR_NONCONVEX_TOL_OPT

Description:

Optimality tolerance used by the nonconvex optimizer.

Possible Values:

Any number between 0.0 and +inf.

Default value:

1.0e-7

- `optimizer_max_time`

Corresponding constant:

MSK_DPAR_OPTIMIZER_MAX_TIME

Description:

Maximum amount of time the optimizer is allowed to spent on the optimization.
A negative number means infinity.

Possible Values:

Any number between -inf and +inf.

Default value:

-1.0

- `presolve_tol_aij`

Corresponding constant:

MSK_DPAR_PREOLVE_TOL_AIJ

Description:

Absolute zero tolerance employed for a_{ij} in the presolve.

Possible Values:

Any number between 1.0e-15 and +inf.

Default value:

1.0e-12

- `presolve_tol_lin_dep`

Corresponding constant:`MSK_DPAR_PRESOLVE_TOL_LIN_DEP`**Description:**

Controls when a constraint is determined to be linearly dependent.

Possible Values:

Any number between 0.0 and +inf.

Default value:`1.0e-6`

- `presolve_tol_s`

Corresponding constant:`MSK_DPAR_PRESOLVE_TOL_S`**Description:**

Absolute zero tolerance employed for s_i in the presolve.

Possible Values:

Any number between 0.0 and +inf.

Default value:`1.0e-8`

- `presolve_tol_x`

Corresponding constant:`MSK_DPAR_PRESOLVE_TOL_X`**Description:**

Absolute zero tolerance employed for x_j in the presolve.

Possible Values:

Any number between 0.0 and +inf.

Default value:`1.0e-8`

- `qcqo_reformulate_rel_drop_tol`

Corresponding constant:`MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL`**Description:**

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

Possible Values:

Any number between 0 and $+\infty$.

Default value:

$1e-15$

- `sim_lu_tol_rel_piv`

Corresponding constant:

`MSK_DPAR_SIM_LU_TOL_REL_PIV`

Description:

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

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

Possible Values:

Any number between $1.0e-6$ and 0.999999 .

Default value:

0.01

- `simplex_abs_tol_piv`

Corresponding constant:

`MSK_DPAR_SIMPLEX_ABS_TOL_PIV`

Description:

Absolute pivot tolerance employed by the simplex optimizers.

Possible Values:

Any number between $1.0e-12$ and $+\infty$.

Default value:

$1.0e-7$

- `upper_obj_cut`

Corresponding constant:

`MSK_DPAR_UPPER_OBJ_CUT`

Description:

If either a primal or dual feasible solution is found proving that the optimal objective value is outside, [`MSK_DPAR_LOWER_OBJ_CUT`, `MSK_DPAR_UPPER_OBJ_CUT`], then MOSEK is terminated.

Possible Values:

Any number between $-\infty$ and $+\infty$.

Default value:

1.0e30

See also:**MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH** Objective bound.

- upper_obj_cut_finite_trh

Corresponding constant:**MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH****Description:**

If the upper objective cut is greater than the value of this value parameter, then the the upper objective cut **MSK_DPAR_UPPER_OBJ_CUT** is treated as ∞ .

Possible Values:

Any number between -inf and +inf.

Default value:

0.5e30

E.3 Integer parameters

- **MSK_IPAR_ALLOC_ADD_QNZ** 305
Controls how the quadratic matrixes are extended.
- **MSK_IPAR_ANA_SOL_BASIS** 305
Controls whether the basis matrix is analyzed in solaution analyzer.
- **MSK_IPAR_ANA_SOL_PRINT_VIOLATED** 306
Controls whether a list of violated constraints is printed.
- **MSK_IPAR_AUTO_SORT_A_BEFORE_OPT** 306
Controls whether the elements in each column of A are sorted before an optimization is performed.
- **MSK_IPAR_AUTO_UPDATE_SOL_INFO** 306
Controls whether the solution information items are automatically updated after an optimization is performed.
- **MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE** 307
Controls the sign of the columns in the basis matrix corresponding to slack variables.
- **MSK_IPAR_BI_CLEAN_OPTIMIZER** 307
Controls which simplex optimizer is used in the clean-up phase.

- **MSK_IPAR_BI_IGNORE_MAX_ITER** 308
Turns on basis identification in case the interior-point optimizer is terminated due to maximum number of iterations.
- **MSK_IPAR_BI_IGNORE_NUM_ERROR** 308
Turns on basis identification in case the interior-point optimizer is terminated due to a numerical problem.
- **MSK_IPAR_BI_MAX_ITERATIONS** 308
Maximum number of iterations after basis identification.
- **MSK_IPAR_CACHE_LICENSE** 309
Control license caching.
- **MSK_IPAR_CACHE_SIZE_L1** 309
Specifies the size of the level 1 cache of the processor.
- **MSK_IPAR_CACHE_SIZE_L2** 309
Specifies the size of the level 2 cache of the processor.
- **MSK_IPAR_CHECK_CONVEXITY** 310
Specify the level of convexity check on quadratic problems
- **MSK_IPAR_CHECK_TASK_DATA** 310
If this feature is turned on, then the task data is checked for bad values i.e. NaNs. before an optimization is performed.
- **MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS** 310
The maximum number of simultaneous optimizations that will be started by the concurrent optimizer.
- **MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX** 311
Priority of the dual simplex algorithm when selecting solvers for concurrent optimization.
- **MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX** 311
Priority of the free simplex optimizer when selecting solvers for concurrent optimization.
- **MSK_IPAR_CONCURRENT_PRIORITY_INTPNT** 311
Priority of the interior-point algorithm when selecting solvers for concurrent optimization.
- **MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX** 312
Priority of the primal simplex algorithm when selecting solvers for concurrent optimization.

- **MSK_IPAR_CPU_TYPE** 312
Specifies the CPU type.
- **MSK_IPAR_DATA_CHECK** 312
Enable data checking for debug purposes.
- **MSK_IPAR_FEASREPAIR_OPTIMIZE** 313
Controls which type of feasibility analysis is to be performed.
- **MSK_IPAR_INFEAS_GENERIC_NAMES** 313
Controls the contents of the infeasibility report.
- **MSK_IPAR_INFEAS_PREFER_PRIMAL** 313
Controls which certificate is used if both primal- and dual- certificate of infeasibility is available.
- **MSK_IPAR_INFEAS_REPORT_AUTO** 314
Turns the feasibility report on or off.
- **MSK_IPAR_INFEAS_REPORT_LEVEL** 314
Controls the contents of the infeasibility report.
- **MSK_IPAR_INTPNT_BASIS** 314
Controls whether basis identification is performed.
- **MSK_IPAR_INTPNT_DIFF_STEP** 315
Controls whether different step sizes are allowed in the primal and dual space.
- **MSK_IPAR_INTPNT_FACTOR_DEBUG_LVL** 315
Controls factorization debug level.
- **MSK_IPAR_INTPNT_FACTOR_METHOD** 316
Controls the method used to factor the Newton equation system.
- **MSK_IPAR_INTPNT_MAX_ITERATIONS** 316
Controls the maximum number of iterations allowed in the interior-point optimizer.
- **MSK_IPAR_INTPNT_MAX_NUM_COR** 316
Maximum number of correction steps.
- **MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS** 316
Maximum number of steps to be used by the iterative search direction refinement.
- **MSK_IPAR_INTPNT_NUM_THREADS** 317
Controls the number of threads employed by the interior-point optimizer. If set to a positive number MOSEK will use this number of threads. If zero the number of threads used will equal the number of cores detected on the machine.

- **MSK_IPAR_INTPNT_OFF_COL_TRH** 317
Controls the aggressiveness of the offending column detection.
- **MSK_IPAR_INTPNT_ORDER_METHOD** 317
Controls the ordering strategy.
- **MSK_IPAR_INTPNT_REGULARIZATION_USE** 318
Controls whether regularization is allowed.
- **MSK_IPAR_INTPNT_SCALING** 318
Controls how the problem is scaled before the interior-point optimizer is used.
- **MSK_IPAR_INTPNT_SOLVE_FORM** 319
Controls whether the primal or the dual problem is solved.
- **MSK_IPAR_INTPNT_STARTING_POINT** 319
Starting point used by the interior-point optimizer.
- **MSK_IPAR_LIC_TRH_EXPIRY_WRN** 319
Controls when expiry warnings are issued.
- **MSK_IPAR_LICENSE_ALLOW_OVERUSE** 320
Controls if license overuse is allowed when caching licenses
- **MSK_IPAR_LICENSE_CACHE_TIME** 320
Setting this parameter no longer has any effect.
- **MSK_IPAR_LICENSE_CHECK_TIME** 320
Controls the license manager client behavior.
- **MSK_IPAR_LICENSE_DEBUG** 321
Controls the license manager client debugging behavior.
- **MSK_IPAR_LICENSE_PAUSE_TIME** 321
Controls license manager client behavior.
- **MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS** 321
Controls license manager client behavior.
- **MSK_IPAR_LICENSE_WAIT** 322
Controls if MOSEK should queue for a license if none is available.
- **MSK_IPAR_LOG** 322
Controls the amount of log information.

- **MSK_IPAR_LOG_BI** 322
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** 323
Controls the logging frequency.
- **MSK_IPAR_LOG_CHECK_CONVEXITY** 323
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.
- **MSK_IPAR_LOG_CONCURRENT** 323
Controls amount of output printed by the concurrent optimizer.
- **MSK_IPAR_LOG_CUT_SECOND_OPT** 324
Controls the reduction in the log levels for the second and any subsequent optimizations.
- **MSK_IPAR_LOG_FACTOR** 324
If turned on, then the factor log lines are added to the log.
- **MSK_IPAR_LOG_FEASREPAIR** 324
Controls the amount of output printed when performing feasibility repair.
- **MSK_IPAR_LOG_FILE** 325
If turned on, then some log info is printed when a file is written or read.
- **MSK_IPAR_LOG_HEAD** 325
If turned on, then a header line is added to the log.
- **MSK_IPAR_LOG_INFEAS_ANA** 325
Controls log level for the infeasibility analyzer.
- **MSK_IPAR_LOG_INTPNT** 326
Controls the amount of log information from the interior-point optimizers.
- **MSK_IPAR_LOG_MIO** 326
Controls the amount of log information from the mixed-integer optimizers.
- **MSK_IPAR_LOG_MIO_FREQ** 326
The mixed-integer solver logging frequency.
- **MSK_IPAR_LOG_NONCONVEX** 326
Controls amount of output printed by the nonconvex optimizer.

- **MSK_IPAR_LOG_OPTIMIZER** 327
Controls the amount of general optimizer information that is logged.
- **MSK_IPAR_LOG_ORDER** 327
If turned on, then factor lines are added to the log.
- **MSK_IPAR_LOG_PARAM** 327
Controls the amount of information printed out about parameter changes.
- **MSK_IPAR_LOG_PRESOLVE** 328
Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.
- **MSK_IPAR_LOG_RESPONSE** 328
Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.
- **MSK_IPAR_LOG_SENSITIVITY** 328
Control logging in sensitivity analyzer.
- **MSK_IPAR_LOG_SENSITIVITY_OPT** 328
Control logging in sensitivity analyzer.
- **MSK_IPAR_LOG_SIM** 329
Controls the amount of log information from the simplex optimizers.
- **MSK_IPAR_LOG_SIM_FREQ** 329
Controls simplex logging frequency.
- **MSK_IPAR_LOG_SIM_MINOR** 329
Currently not in use.
- **MSK_IPAR_LOG_SIM_NETWORK_FREQ** 330
Controls the network simplex logging frequency.
- **MSK_IPAR_LOG_STORAGE** 330
Controls the memory related log information.
- **MSK_IPAR_LP_WRITE_IGNORE_INCOMPATIBLE_ITEMS** 330
Controls the result of writing a problem containing incompatible items to an LP file.
- **MSK_IPAR_MAX_NUM_WARNINGS** 331
Warning level. A higher value results in more warnings.
- **MSK_IPAR_MIO_BRANCH_DIR** 331
Controls whether the mixed-integer optimizer is branching up or down by default.

- **MSK_IPAR_MIO_BRANCH_PRIORITIES_USE** 331
Controls whether branching priorities are used by the mixed-integer optimizer.
- **MSK_IPAR_MIO_CONSTRUCT_SOL** 332
Controls if an initial mixed integer solution should be constructed from the values of the integer variables.
- **MSK_IPAR_MIO_CONT_SOL** 332
Controls the meaning of interior-point and basic solutions in mixed integer problems.
- **MSK_IPAR_MIO_CUT_LEVEL_ROOT** 333
Controls the cut level employed by the mixed-integer optimizer at the root node.
- **MSK_IPAR_MIO_CUT_LEVEL_TREE** 333
Controls the cut level employed by the mixed-integer optimizer in the tree.
- **MSK_IPAR_MIO_FEASPUMP_LEVEL** 333
Controls the feasibility pump heuristic which is used to construct a good initial feasible solution.
- **MSK_IPAR_MIO_HEURISTIC_LEVEL** 334
Controls the heuristic employed by the mixed-integer optimizer to locate an initial integer feasible solution.
- **MSK_IPAR_MIO_HOTSTART** 334
Controls whether the integer optimizer is hot-started.
- **MSK_IPAR_MIO_KEEP_BASIS** 334
Controls whether the integer presolve keeps bases in memory.
- **MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER** 335
Controls the size of the local search space when doing local branching.
- **MSK_IPAR_MIO_MAX_NUM_BRANCHES** 335
Maximum number of branches allowed during the branch and bound search.
- **MSK_IPAR_MIO_MAX_NUM_RELAXS** 335
Maximum number of relaxations in branch and bound search.
- **MSK_IPAR_MIO_MAX_NUM_SOLUTIONS** 336
Controls how many feasible solutions the mixed-integer optimizer investigates.
- **MSK_IPAR_MIO_MODE** 336
Turns on/off the mixed-integer mode.

- **MSK_IPAR_MIO_NODE_OPTIMIZER**.....337
Controls which optimizer is employed at the non-root nodes in the mixed-integer optimizer.
- **MSK_IPAR_MIO_NODE_SELECTION**.....337
Controls the node selection strategy employed by the mixed-integer optimizer.
- **MSK_IPAR_MIO_OPTIMIZER_MODE**.....338
An experimental feature.
- **MSK_IPAR_MIO_PRESOLVE_AGGREGATE**.....338
Controls whether problem aggregation is performed in the mixed-integer presolve.
- **MSK_IPAR_MIO_PRESOLVE_PROBING**.....338
Controls whether probing is employed by the mixed-integer presolve.
- **MSK_IPAR_MIO_PRESOLVE_USE**.....339
Controls whether presolve is performed by the mixed-integer optimizer.
- **MSK_IPAR_MIO_ROOT_OPTIMIZER**.....339
Controls which optimizer is employed at the root node in the mixed-integer optimizer.
- **MSK_IPAR_MIO_STRONG_BRANCH**.....340
The depth from the root in which strong branching is employed.
- **MSK_IPAR_NONCONVEX_MAX_ITERATIONS**.....340
Maximum number of iterations that can be used by the nonconvex optimizer.
- **MSK_IPAR_OBJECTIVE_SENSE**.....340
If the objective sense for the task is undefined, then the value of this parameter is used as the default objective sense.
- **MSK_IPAR_OPF_MAX_TERMS_PER_LINE**.....341
The maximum number of terms (linear and quadratic) per line when an OPF file is written.
- **MSK_IPAR_OPF_WRITE_HEADER**.....341
Write a text header with date and MOSEK version in an OPF file.
- **MSK_IPAR_OPF_WRITE_HINTS**.....341
Write a hint section with problem dimensions in the beginning of an OPF file.
- **MSK_IPAR_OPF_WRITE_PARAMETERS**.....341
Write a parameter section in an OPF file.

- **MSK_IPAR_OPF_WRITE_PROBLEM** 342
Write objective, constraints, bounds etc. to an OPF file.
- **MSK_IPAR_OPF_WRITE_SOL_BAS** 342
Controls what is written to the OPF files.
- **MSK_IPAR_OPF_WRITE_SOL_ITG** 342
Controls what is written to the OPF files.
- **MSK_IPAR_OPF_WRITE_SOL_ITR** 343
Controls what is written to the OPF files.
- **MSK_IPAR_OPF_WRITE_SOLUTIONS** 343
Enable inclusion of solutions in the OPF files.
- **MSK_IPAR_OPTIMIZER** 343
Controls which optimizer is used to optimize the task.
- **MSK_IPAR_PARAM_READ_CASE_NAME** 344
If turned on, then names in the parameter file are case sensitive.
- **MSK_IPAR_PARAM_READ_IGN_ERROR** 344
If turned on, then errors in paramter settings is ignored.
- **MSK_IPAR_PRESOLVE_ELIM_FILL** 345
Maximum amount of fill-in in the elimination phase.
- **MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES** 345
Control the maximum number of times the eliminator is tried.
- **MSK_IPAR_PRESOLVE_ELIMINATOR_USE** 345
Controls whether free or implied free variables are eliminated from the problem.
- **MSK_IPAR_PRESOLVE_LEVEL** 346
Currently not used.
- **MSK_IPAR_PRESOLVE_LINDEP_USE** 346
Controls whether the linear constraints are checked for linear dependencies.
- **MSK_IPAR_PRESOLVE_LINDEP_WORK_LIM** 346
Controls linear dependency check in presolve.
- **MSK_IPAR_PRESOLVE_USE** 346
Controls whether the presolve is applied to a problem before it is optimized.
- **MSK_IPAR_QO_SEPARABLE_REFORMULATION** 347
Determine if Quadratic programing problems should be reformulated to separable form.

- **MSK_IPAR_READ_ADD_ANZ** 347
Controls how the constraint matrix is extended.
- **MSK_IPAR_READ_ADD_CON** 347
Additional number of constraints that is made room for in the problem.
- **MSK_IPAR_READ_ADD_CONE** 348
Additional number of conic constraints that is made room for in the problem.
- **MSK_IPAR_READ_ADD_QNZ** 348
Controls how the quadratic matrixes are extended.
- **MSK_IPAR_READ_ADD_VAR** 348
Additional number of variables that is made room for in the problem.
- **MSK_IPAR_READ_ANZ** 348
Controls the expected number of constraint non-zeros.
- **MSK_IPAR_READ_CON** 349
Controls the expected number of constraints.
- **MSK_IPAR_READ_CONE** 349
Controls the expected number of conic constraints.
- **MSK_IPAR_READ_DATA_COMPRESSED** 349
Controls the input file decompression.
- **MSK_IPAR_READ_DATA_FORMAT** 350
Format of the data file to be read.
- **MSK_IPAR_READ_KEEP_FREE_CON** 350
Controls whether the free constraints are included in the problem.
- **MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU** 350
Controls how the LP files are interpreted.
- **MSK_IPAR_READ_LP_QUOTED_NAMES** 351
If a name is in quotes when reading an LP file, the quotes will be removed.
- **MSK_IPAR_READ_MPS_FORMAT** 351
Controls how strictly the MPS file reader interprets the MPS format.
- **MSK_IPAR_READ_MPS_KEEP_INT** 352
Controls if integer constraints are read.
- **MSK_IPAR_READ_MPS_OBJ_SENSE** 352
Controls the MPS format extensions.

- **MSK_IPAR_READ_MPS_QUOTED_NAMES** 352
Controls the MPS format extensions.
- **MSK_IPAR_READ_MPS_RELAX** 352
Controls the meaning of integer constraints.
- **MSK_IPAR_READ_MPS_WIDTH** 353
Controls the maximal number of characters allowed in one line of the MPS file.
- **MSK_IPAR_READ_Q_MODE** 353
Controls how the Q matrices are read from the MPS file.
- **MSK_IPAR_READ_QNZ** 353
Controls the expected number of quadratic non-zeros.
- **MSK_IPAR_READ_TASK_IGNORE_PARAM** 354
Controls what information is used from the task files.
- **MSK_IPAR_READ_VAR** 354
Controls the expected number of variables.
- **MSK_IPAR_SENSITIVITY_OPTIMIZER** 354
Controls which optimizer is used for optimal partition sensitivity analysis.
- **MSK_IPAR_SENSITIVITY_TYPE** 355
Controls which type of sensitivity analysis is to be performed.
- **MSK_IPAR_SIM_BASIS_FACTOR_USE** 355
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 penalty.
- **MSK_IPAR_SIM_DEGEN** 356
Controls how aggressively degeneration is handled.
- **MSK_IPAR_SIM_DUAL_CRASH** 356
Controls whether crashing is performed in the dual simplex optimizer.
- **MSK_IPAR_SIM_DUAL_PHASEONE_METHOD** 356
An experimental feature.
- **MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION** 357
Controls how aggressively restricted selection is used.
- **MSK_IPAR_SIM_DUAL_SELECTION** 357
Controls the dual simplex strategy.

• MSK_IPAR_SIM_EXPLOIT_DUPVEC	358
Controls if the simplex optimizers are allowed to exploit duplicated columns.	
• MSK_IPAR_SIM_HOTSTART	358
Controls the type of hot-start that the simplex optimizer perform.	
• MSK_IPAR_SIM_HOTSTART_LU	358
Determines if the simplex optimizer should exploit the initial factorization.	
• MSK_IPAR_SIM_INTEGER	359
An experimental feature.	
• MSK_IPAR_SIM_MAX_ITERATIONS	359
Maximum number of iterations that can be used by a simplex optimizer.	
• MSK_IPAR_SIM_MAX_NUM_SETBACKS	359
Controls how many set-backs that are allowed within a simplex optimizer.	
• MSK_IPAR_SIM_NETWORK_DETECT	360
Level of aggressiveness of network detection.	
• MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART	360
Level of aggressiveness of network detection in a simplex hot-start.	
• MSK_IPAR_SIM_NETWORK_DETECT_METHOD	360
Controls which type of detection method the network extraction should use.	
• MSK_IPAR_SIM_NON_SINGULAR	361
Controls if the simplex optimizer ensures a non-singular basis, if possible.	
• MSK_IPAR_SIM_PRIMAL_CRASH	361
Controls the simplex crash.	
• MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD	361
An experimental feature.	
• MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION	362
Controls how aggressively restricted selection is used.	
• MSK_IPAR_SIM_PRIMAL_SELECTION	362
Controls the primal simplex strategy.	
• MSK_IPAR_SIM_REFACTOR_FREQ	363
Controls the basis refactoring frequency.	
• MSK_IPAR_SIM_REFORMULATION	363
Controls if the simplex optimizers are allowed to reformulate the problem.	

- **MSK_IPAR_SIM_SAVE_LU** 363
Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.
- **MSK_IPAR_SIM_SCALING** 364
Controls how much effort is used in scaling the problem before a simplex optimizer is used.
- **MSK_IPAR_SIM_SCALING_METHOD** 364
Controls how the problem is scaled before a simplex optimizer is used.
- **MSK_IPAR_SIM_SOLVE_FORM** 364
Controls whether the primal or the dual problem is solved by the primal-/dual- simplex optimizer.
- **MSK_IPAR_SIM_STABILITY_PRIORITY** 365
Controls how high priority the numerical stability should be given.
- **MSK_IPAR_SIM_SWITCH_OPTIMIZER** 365
Controls the simplex behavior.
- **MSK_IPAR_SOL_FILTER_KEEP_BASIC** 366
Controls the license manager client behavior.
- **MSK_IPAR_SOL_FILTER_KEEP_RANGED** 366
Control the contents of the solution files.
- **MSK_IPAR_SOL_QUOTED_NAMES** 366
Controls the solution file format.
- **MSK_IPAR_SOL_READ_NAME_WIDTH** 367
Controls the input solution file format.
- **MSK_IPAR_SOL_READ_WIDTH** 367
Controls the input solution file format.
- **MSK_IPAR_SOLUTION_CALLBACK** 367
Indicates whether solution call-backs will be performed during the optimization.
- **MSK_IPAR_TIMING_LEVEL** 368
Controls the a amount of timing performed inside MOSEK.
- **MSK_IPAR_WARNING_LEVEL** 368
Warning level.

- **MSK_IPAR_WRITE_BAS_CONSTRAINTS** 368
Controls the basic solution file format.
- **MSK_IPAR_WRITE_BAS_HEAD** 368
Controls the basic solution file format.
- **MSK_IPAR_WRITE_BAS_VARIABLES** 369
Controls the basic solution file format.
- **MSK_IPAR_WRITE_DATA_COMPRESSED** 369
Controls output file compression.
- **MSK_IPAR_WRITE_DATA_FORMAT** 369
Controls the output file format.
- **MSK_IPAR_WRITE_DATA_PARAM** 370
Controls output file data.
- **MSK_IPAR_WRITE_FREE_CON** 370
Controls the output file data.
- **MSK_IPAR_WRITE_GENERIC_NAMES** 370
Controls the output file data.
- **MSK_IPAR_WRITE_GENERIC_NAMES_IO** 371
Index origin used in generic names.
- **MSK_IPAR_WRITE_INT_CONSTRAINTS** 371
Controls the integer solution file format.
- **MSK_IPAR_WRITE_INT_HEAD** 371
Controls the integer solution file format.
- **MSK_IPAR_WRITE_INT_VARIABLES** 372
Controls the integer solution file format.
- **MSK_IPAR_WRITE_LP_LINE_WIDTH** 372
Controls the LP output file format.
- **MSK_IPAR_WRITE_LP_QUOTED_NAMES** 372
Controls LP output file format.
- **MSK_IPAR_WRITE_LP_STRICT_FORMAT** 373
Controls whether LP output files satisfy the LP format strictly.
- **MSK_IPAR_WRITE_LP_TERMS_PER_LINE** 373
Controls the LP output file format.

- **MSK_IPAR_WRITE_MPS_INT** 373
Controls the output file data.
- **MSK_IPAR_WRITE_MPS_OBJ_SENSE** 373
Controls the output file data.
- **MSK_IPAR_WRITE_MPS_QUOTED_NAMES** 374
Controls the output file data.
- **MSK_IPAR_WRITE_MPS_STRICT** 374
Controls the output MPS file format.
- **MSK_IPAR_WRITE_PRECISION** 374
Controls data precision employed in when writing an MPS file.
- **MSK_IPAR_WRITE_SOL_CONSTRAINTS** 375
Controls the solution file format.
- **MSK_IPAR_WRITE_SOL_HEAD** 375
Controls solution file format.
- **MSK_IPAR_WRITE_SOL_VARIABLES** 375
Controls the solution file format.
- **MSK_IPAR_WRITE_TASK_INC_SOL** 376
Controls whether the solutions are stored in the task file too.
- **MSK_IPAR_WRITE_XML_MODE** 376
Controls if linear coefficients should be written by row or column when writing in the XML file format.

- **alloc_add_qnz**

Corresponding constant:

MSK_IPAR_ALLOC_ADD_QNZ

Description:

Additional number of Q non-zeros that are allocated space for when **numanz** exceeds **maxnumqnz** during addition of new Q entries.

Possible Values:

Any number between 0 and +inf.

Default value:

5000

- **ana_sol_basis**

Corresponding constant:

MSK_IPAR_ANA_SOL_BASIS

Description:

Controls whether the basis matrix is analyzed in solution analyzer.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

• **ana_sol_print_violated****Corresponding constant:**

MSK_IPAR_ANA_SOL_PRINT_VIOLATED

Description:**Possible values:**

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

• **auto_sort_a_before_opt****Corresponding constant:**

MSK_IPAR_AUTO_SORT_A_BEFORE_OPT

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

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

• **auto_update_sol_info****Corresponding constant:**

MSK_IPAR_AUTO_UPDATE_SOL_INFO

Description:

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

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `basis_solve_use_plus_one`

Corresponding constant:

MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE

Description:

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

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- `bi_clean_optimizer`

Corresponding constant:

MSK_IPAR_BI_CLEAN_OPTIMIZER

Description:

Controls which simplex optimizer is used in the clean-up phase.

Possible values:

MSK_OPTIMIZER_INTPNT The interior-point optimizer is used.

MSK_OPTIMIZER_CONCURRENT The optimizer for nonconvex nonlinear problems.

MSK_OPTIMIZER_MIXED_INT The mixed-integer optimizer.

MSK_OPTIMIZER_DUAL_SIMPLEX The dual simplex optimizer is used.

MSK_OPTIMIZER_FREE The optimizer is chosen automatically.

MSK_OPTIMIZER_PRIMAL_DUAL_SIMPLEX The primal dual simplex optimizer is used.

MSK_OPTIMIZER_CONIC The optimizer for problems having conic constraints.

MSK_OPTIMIZER_NONCONVEX The optimizer for nonconvex nonlinear problems.

MSK_OPTIMIZER_QCONE For internal use only.

MSK_OPTIMIZER_PRIMAL_SIMPLEX The primal simplex optimizer is used.

MSK_OPTIMIZER_FREE_SIMPLEX One of the simplex optimizers is used.

Default value:

MSK_OPTIMIZER_FREE

- `bi_ignore_max_iter`

Corresponding constant:

MSK_IPAR_BI_IGNORE_MAX_ITER

Description:

If the parameter `MSK_IPAR_INTPNT_BASIS` has the value `MSK_BI_NO_ERROR` and the interior-point optimizer has terminated due to maximum number of iterations, then basis identification is performed if this parameter has the value `MSK_ON`.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- `bi_ignore_num_error`

Corresponding constant:

MSK_IPAR_BI_IGNORE_NUM_ERROR

Description:

If the parameter `MSK_IPAR_INTPNT_BASIS` has the value `MSK_BI_NO_ERROR` and the interior-point optimizer has terminated due to a numerical problem, then basis identification is performed if this parameter has the value `MSK_ON`.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- `bi_max_iterations`

Corresponding constant:

MSK_IPAR_BI_MAX_ITERATIONS

Description:

Controls the maximum number of simplex iterations allowed to optimize a basis after the basis identification.

Possible Values:

Any number between 0 and +inf.

Default value:

1000000

- `cache_license`

Corresponding constant:

MSK_IPAR_CACHE_LICENSE

Description:

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

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

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `cache_size_l1`

Corresponding constant:

MSK_IPAR_CACHE_SIZE_L1

Description:

Specifies the size of the cache of the computer. This parameter is potentially very important for the efficiency on computers if MOSEK cannot determine the cache size automatically. If the cache size is negative, then MOSEK tries to determine the value automatically.

Possible Values:

Any number between -inf and +inf.

Default value:

-1

- `cache_size_l2`

Corresponding constant:

MSK_IPAR_CACHE_SIZE_L2

Description:

Specifies the size of the cache of the computer. This parameter is potentially very important for the efficiency on computers where MOSEK cannot determine the cache size automatically. If the cache size is negative, then MOSEK tries to determine the value automatically.

Possible Values:

Any number between -inf and +inf.

Default value:

-1

- `check_convexity`

Corresponding constant:

`MSK_IPAR_CHECK_CONVEXITY`

Description:

Specify the level of convexity check on quadratic problems

Possible values:

`MSK_CHECK_CONVEXITY_SIMPLE` Perform simple and fast convexity check.

`MSK_CHECK_CONVEXITY_NONE` No convexity check.

`MSK_CHECK_CONVEXITY_FULL` Perform a full convexity check.

Default value:

`MSK_CHECK_CONVEXITY_FULL`

- `check_task_data`

Corresponding constant:

`MSK_IPAR_CHECK_TASK_DATA`

Description:

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

Possible values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

`MSK_ON`

- `concurrent_num_optimizers`

Corresponding constant:

`MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS`

Description:

The maximum number of simultaneous optimizations that will be started by the concurrent optimizer.

Possible Values:

Any number between 0 and +inf.

Default value:

2

- `concurrent_priority_dual_simplex`

Corresponding constant:

`MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX`

Description:

Priority of the dual simplex algorithm when selecting solvers for concurrent optimization.

Possible Values:

Any number between 0 and +inf.

Default value:

2

- `concurrent_priority_free_simplex`

Corresponding constant:

`MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX`

Description:

Priority of the free simplex optimizer when selecting solvers for concurrent optimization.

Possible Values:

Any number between 0 and +inf.

Default value:

3

- `concurrent_priority_intpnt`

Corresponding constant:

`MSK_IPAR_CONCURRENT_PRIORITY_INTPNT`

Description:

Priority of the interior-point algorithm when selecting solvers for concurrent optimization.

Possible Values:

Any number between 0 and +inf.

Default value:

4

- `concurrent_priority_primal_simplex`

Corresponding constant:

MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX

Description:

Priority of the primal simplex algorithm when selecting solvers for concurrent optimization.

Possible Values:

Any number between 0 and +inf.

Default value:

1

- `cpu_type`

Corresponding constant:

MSK_IPAR_CPU_TYPE

Description:

Specifies the CPU type. By default MOSEK tries to auto detect the CPU type. Therefore, we recommend to change this parameter only if the auto detection does not work properly.

Possible values:

MSK_CPU_POWERPC_G5 A G5 PowerPC CPU.
 MSK_CPU_INTEL_PM An Intel PM cpu.
 MSK_CPU_GENERIC An generic CPU type for the platform
 MSK_CPU_UNKNOWN An unknown CPU.
 MSK_CPU_AMD_OPTERON An AMD Opteron (64 bit).
 MSK_CPU_INTEL_ITANIUM2 An Intel Itanium2.
 MSK_CPU_AMD_ATHLON An AMD Athlon.
 MSK_CPU_HP_PARISC20 An HP PA RISC version 2.0 CPU.
 MSK_CPU_INTEL_P4 An Intel Pentium P4 or Intel Xeon.
 MSK_CPU_INTEL_P3 An Intel Pentium P3.
 MSK_CPU_INTEL_CORE2 An Intel CORE2 cpu.

Default value:

MSK_CPU_UNKNOWN

- `data_check`

Corresponding constant:

MSK_IPAR_DATA_CHECK

Description:

If this option is turned on, then extensive data checking is enabled. It will slow down MOSEK but on the other hand help locating bugs.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `feasrepair_optimize`

Corresponding constant:

MSK_IPAR_FEASREPAIR_OPTIMIZE

Description:

Controls which type of feasibility analysis is to be performed.

Possible values:

MSK_FEASREPAIR_OPTIMIZE_NONE Do not optimize the feasibility repair problem.

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

MSK_FEASREPAIR_OPTIMIZE_PENALTY Minimize weighted sum of violations.

Default value:

MSK_FEASREPAIR_OPTIMIZE_NONE

- `infeas_generic_names`

Corresponding constant:

MSK_IPAR_INFEAS_GENERIC_NAMES

Description:

Controls whether generic names are used when an infeasible subproblem is created.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- `infeas_prefer_primal`

Corresponding constant:

MSK_IPAR_INFEAS_PREFER_PRIMAL

Description:

If both certificates of primal and dual infeasibility are supplied then only the primal is used when this option is turned on.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `infeas_report_auto`

Corresponding constant:

MSK_IPAR_INFEAS_REPORT_AUTO

Description:

Controls whether an infeasibility report is automatically produced after the optimization if the problem is primal or dual infeasible.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- `infeas_report_level`

Corresponding constant:

MSK_IPAR_INFEAS_REPORT_LEVEL

Description:

Controls the amount of information presented in an infeasibility report. Higher values imply more information.

Possible Values:

Any number between 0 and +inf.

Default value:

1

- `intpnt_basis`

Corresponding constant:

MSK_IPAR_INTPNT_BASIS

Description:

Controls whether the interior-point optimizer also computes an optimal basis.

Possible values:

`MSK_BI_ALWAYS` Basis identification is always performed even if the interior-point optimizer terminates abnormally.

`MSK_BI_NO_ERROR` Basis identification is performed if the interior-point optimizer terminates without an error.

`MSK_BI_NEVER` Never do basis identification.

`MSK_BI_IF_FEASIBLE` Basis identification is not performed if the interior-point optimizer terminates with a problem status saying that the problem is primal or dual infeasible.

`MSK_BI_OTHER` Try another BI method.

Default value:

`MSK_BI_ALWAYS`

See also:

`MSK_IPAR_BI_IGNORE_MAX_ITER` Turns on basis identification in case the interior-point optimizer is terminated due to maximum number of iterations.

`MSK_IPAR_BI_IGNORE_NUM_ERROR` Turns on basis identification in case the interior-point optimizer is terminated due to a numerical problem.

- `intpnt_diff_step`

Corresponding constant:

`MSK_IPAR_INTPNT_DIFF_STEP`

Description:

Controls whether different step sizes are allowed in the primal and dual space.

Possible values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

`MSK_ON`

- `intpnt_factor_debug_lvl`

Corresponding constant:

`MSK_IPAR_INTPNT_FACTOR_DEBUG_LVL`

Description:

Controls factorization debug level.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

0

- `intpnt_factor_method`

Corresponding constant:

`MSK_IPAR_INTPNT_FACTOR_METHOD`

Description:

Controls the method used to factor the Newton equation system.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

0

- `intpnt_max_iterations`

Corresponding constant:

`MSK_IPAR_INTPNT_MAX_ITERATIONS`

Description:

Controls the maximum number of iterations allowed in the interior-point optimizer.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

400

- `intpnt_max_num_cor`

Corresponding constant:

`MSK_IPAR_INTPNT_MAX_NUM_COR`

Description:

Controls the maximum number of correctors allowed by the multiple corrector procedure. A negative value means that MOSEK is making the choice.

Possible Values:

Any number between -1 and $+\infty$.

Default value:

-1

- `intpnt_max_num_refinement_steps`

Corresponding constant:

MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS

Description:

Maximum number of steps to be used by the iterative refinement of the search direction. A negative value implies that the optimizer Chooses the maximum number of iterative refinement steps.

Possible Values:

Any number between -inf and +inf.

Default value:

-1

- `intpnt_num_threads`

Corresponding constant:

MSK_IPAR_INTPNT_NUM_THREADS

Description:

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

Possible Values:

Any integer greater or equal to 0.

Default value:

1

- `intpnt_off_col_trh`

Corresponding constant:

MSK_IPAR_INTPNT_OFF_COL_TRH

Description:

Controls how many offending columns are detected in the Jacobian of the constraint matrix.

1 means aggressive detection, higher values mean less aggressive detection.

0 means no detection.

Possible Values:

Any number between 0 and +inf.

Default value:

40

- `intpnt_order_method`

Corresponding constant:

MSK_IPAR_INTPNT_ORDER_METHOD

Description:

Controls the ordering strategy used by the interior-point optimizer when factorizing the Newton equation system.

Possible values:

MSK_ORDER_METHOD_NONE No ordering is used.

MSK_ORDER_METHOD_APPMINLOC2 A variant of the approximate minimum local-fill-in ordering is used.

MSK_ORDER_METHOD_APPMINLOC1 Approximate minimum local-fill-in ordering is used.

MSK_ORDER_METHOD_GRAPHPAR2 An alternative graph partitioning based ordering.

MSK_ORDER_METHOD_FREE The ordering method is chosen automatically.

MSK_ORDER_METHOD_GRAPHPAR1 Graph partitioning based ordering.

Default value:

MSK_ORDER_METHOD_FREE

- intpnt_regularization_use

Corresponding constant:

MSK_IPAR_INTPNT_REGULARIZATION_USE

Description:

Controls whether regularization is allowed.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- intpnt_scaling

Corresponding constant:

MSK_IPAR_INTPNT_SCALING

Description:

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

Possible values:

MSK_SCALING_NONE No scaling is performed.

MSK_SCALING_MODERATE A conservative scaling is performed.

MSK_SCALING_AGGRESSIVE A very aggressive scaling is performed.

MSK_SCALING_FREE The optimizer chooses the scaling heuristic.

Default value:

MSK_SCALING_FREE

- `intpnt_solve_form`

Corresponding constant:

MSK_IPAR_INTPNT_SOLVE_FORM

Description:

Controls whether the primal or the dual problem is solved.

Possible values:

MSK_SOLVE_PRIMAL The optimizer should solve the primal problem.

MSK_SOLVE_DUAL The optimizer should solve the dual problem.

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

Default value:

MSK_SOLVE_FREE

- `intpnt_starting_point`

Corresponding constant:

MSK_IPAR_INTPNT_STARTING_POINT

Description:

Starting point used by the interior-point optimizer.

Possible values:

MSK_STARTING_POINT_GUESS The optimizer guesses a starting point.

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

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

MSK_STARTING_POINT_FREE The starting point is chosen automatically.

Default value:

MSK_STARTING_POINT_FREE

- `lic_trh_expiry_wrn`

Corresponding constant:

MSK_IPAR_LIC_TRH_EXPIRY_WRN

Description:

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

Possible Values:

Any number between 0 and +inf.

Default value:

7

- `license_allow_overuse`

Corresponding constant:

MSK_IPAR_LICENSE_ALLOW_OVERUSE

Description:

Controls if license overuse is allowed when caching licenses

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `license_cache_time`

Corresponding constant:

MSK_IPAR_LICENSE_CACHE_TIME

Description:

Setting this parameter no longer has any effect. Please see [MSK_IPAR_CACHE_LICENSE](#) for an alternative.

Possible Values:

Any number between 0 and 65555.

Default value:

5

- `license_check_time`

Corresponding constant:

MSK_IPAR_LICENSE_CHECK_TIME

Description:

The parameter specifies the number of seconds between the checks of all the active licenses in the MOSEK environment license cache. These checks are performed to determine if the licenses should be returned to the server.

Possible Values:

Any number between 1 and 120.

Default value:

1

- `license_debug`

Corresponding constant:

`MSK_IPAR_LICENSE_DEBUG`

Description:

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

Possible values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

`MSK_OFF`

- `license_pause_time`

Corresponding constant:

`MSK_IPAR_LICENSE_PAUSE_TIME`

Description:

If `MSK_IPAR_LICENSE_WAIT=MSK_ON` and no license is available, then MOSEK sleeps a number of milliseconds between each check of whether a license has become free.

Possible Values:

Any number between 0 and 1000000.

Default value:

100

- `license_suppress_expire_wrns`

Corresponding constant:

`MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS`

Description:

Controls whether license features expire warnings are suppressed.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- `license_wait`

Corresponding constant:

MSK_IPAR_LICENSE_WAIT

Description:

If all licenses are in use MOSEK returns with an error code. However, by turning on this parameter MOSEK will wait for an available license.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- `log`

Corresponding constant:

MSK_IPAR_LOG

Description:

Controls the amount of log information. The value 0 implies that all log information is suppressed. A higher level implies that more information is logged.

Please note that if a task is employed to solve a sequence of optimization problems the value of this parameter is reduced by the value of `MSK_IPAR_LOG_CUT_SECOND_OPT` for the second and any subsequent optimizations.

Possible Values:

Any number between 0 and +inf.

Default value:

10

See also:

`MSK_IPAR_LOG_CUT_SECOND_OPT` Controls the reduction in the log levels for the second and any subsequent optimizations.

- `log_bi`

Corresponding constant:

MSK_IPAR_LOG_BI

Description:

Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.

Possible Values:

Any number between 0 and +inf.

Default value:

4

- `log.bi_freq`

Corresponding constant:

MSK_IPAR_LOG_BI_FREQ

Description:

Controls how frequent the optimizer outputs information about the basis identification and how frequent the user-defined call-back function is called.

Possible Values:

Any number between 0 and +inf.

Default value:

2500

- `log.check_convexity`

Corresponding constant:

MSK_IPAR_LOG_CHECK_CONVEXITY

Description:

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

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

Possible Values:

Any number between 0 and +inf.

Default value:

0

- `log.concurrent`

Corresponding constant:

MSK_IPAR_LOG_CONCURRENT

Description:

Controls amount of output printed by the concurrent optimizer.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

1

- `log_cut_second_opt`

Corresponding constant:

`MSK_IPAR_LOG_CUT_SECOND_OPT`

Description:

If a task is employed to solve a sequence of optimization problems, then the value of the log levels is reduced by the value of this parameter. E.g `MSK_IPAR_LOG` and `MSK_IPAR_LOG_SIM` are reduced by the value of this parameter for the second and any subsequent optimizations.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

1

See also:

`MSK_IPAR_LOG` Controls the amount of log information.

`MSK_IPAR_LOG_INTPNT` Controls the amount of log information from the interior-point optimizers.

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

`MSK_IPAR_LOG_SIM` Controls the amount of log information from the simplex optimizers.

- `log_factor`

Corresponding constant:

`MSK_IPAR_LOG_FACTOR`

Description:

If turned on, then the factor log lines are added to the log.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

1

- `log_feasrepair`

Corresponding constant:

MSK_IPAR_LOG_FEASREPAIR

Description:

Controls the amount of output printed when performing feasibility repair.

Possible Values:

Any number between 0 and +inf.

Default value:

0

• log_file

Corresponding constant:

MSK_IPAR_LOG_FILE

Description:

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

Possible Values:

Any number between 0 and +inf.

Default value:

1

• log_head

Corresponding constant:

MSK_IPAR_LOG_HEAD

Description:

If turned on, then a header line is added to the log.

Possible Values:

Any number between 0 and +inf.

Default value:

1

• log_infeas_ana

Corresponding constant:

MSK_IPAR_LOG_INFEAS_ANA

Description:

Controls amount of output printed by the infeasibility analyzer procedures. A higher level implies that more information is logged.

Possible Values:

Any number between 0 and +inf.

Default value:

1

• `log_intpnt`**Corresponding constant:**

MSK_IPAR_LOG_INTPNT

Description:

Controls amount of output printed by the interior-point optimizer. A higher level implies that more information is logged.

Possible Values:

Any number between 0 and +inf.

Default value:

4

• `log_mio`**Corresponding constant:**

MSK_IPAR_LOG_MIO

Description:

Controls the log level for the mixed-integer optimizer. A higher level implies that more information is logged.

Possible Values:

Any number between 0 and +inf.

Default value:

4

• `log_mio_freq`**Corresponding constant:**

MSK_IPAR_LOG_MIO_FREQ

Description:

Controls how frequent the mixed-integer optimizer prints the log line. It will print line every time `MSK_IPAR_LOG_MIO_FREQ` relaxations have been solved.

Possible Values:

A integer value.

Default value:

1000

• `log_nonconvex`

Corresponding constant:

MSK_IPAR_LOG_NONCONVEX

Description:

Controls amount of output printed by the nonconvex optimizer.

Possible Values:

Any number between 0 and +inf.

Default value:

1

• log_optimizer

Corresponding constant:

MSK_IPAR_LOG_OPTIMIZER

Description:

Controls the amount of general optimizer information that is logged.

Possible Values:

Any number between 0 and +inf.

Default value:

1

• log_order

Corresponding constant:

MSK_IPAR_LOG_ORDER

Description:

If turned on, then factor lines are added to the log.

Possible Values:

Any number between 0 and +inf.

Default value:

1

• log_param

Corresponding constant:

MSK_IPAR_LOG_PARAM

Description:

Controls the amount of information printed out about parameter changes.

Possible Values:

Any number between 0 and +inf.

Default value:

0

• log_presolve

Corresponding constant:

MSK_IPAR_LOG_PRESOLVE

Description:

Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.

Possible Values:

Any number between 0 and +inf.

Default value:

1

• log_response

Corresponding constant:

MSK_IPAR_LOG_RESPONSE

Description:

Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.

Possible Values:

Any number between 0 and +inf.

Default value:

0

• log_sensitivity

Corresponding constant:

MSK_IPAR_LOG_SENSITIVITY

Description:

Controls the amount of logging during the sensitivity analysis. 0: Means no logging information is produced. 1: Timing information is printed. 2: Sensitivity results are printed.

Possible Values:

Any number between 0 and +inf.

Default value:

1

• log_sensitivity_opt

Corresponding constant:

MSK_IPAR_LOG_SENSITIVITY_OPT

Description:

Controls the amount of logging from the optimizers employed during the sensitivity analysis. 0 means no logging information is produced.

Possible Values:

Any number between 0 and +inf.

Default value:

0

• log_sim

Corresponding constant:

MSK_IPAR_LOG_SIM

Description:

Controls amount of output printed by the simplex optimizer. A higher level implies that more information is logged.

Possible Values:

Any number between 0 and +inf.

Default value:

4

• log_sim_freq

Corresponding constant:

MSK_IPAR_LOG_SIM_FREQ

Description:

Controls how frequent the simplex optimizer outputs information about the optimization and how frequent the user-defined call-back function is called.

Possible Values:

Any number between 0 and +inf.

Default value:

500

• log_sim_minor

Corresponding constant:

MSK_IPAR_LOG_SIM_MINOR

Description:

Currently not in use.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

1

- `log.sim.network.freq`

Corresponding constant:

`MSK_IPAR_LOG_SIM_NETWORK_FREQ`

Description:

Controls how frequent the network simplex optimizer outputs information about the optimization and how frequent the user-defined call-back function is called. The network optimizer will use a logging frequency equal to `MSK_IPAR_LOG_SIM_FREQ` times `MSK_IPAR_LOG_SIM_NETWORK_FREQ`.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

50

- `log.storage`

Corresponding constant:

`MSK_IPAR_LOG_STORAGE`

Description:

When turned on, MOSEK prints messages regarding the storage usage and allocation.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

0

- `lp.write.ignore.incompatible.items`

Corresponding constant:

`MSK_IPAR_LP_WRITE_IGNORE_INCOMPATIBLE_ITEMS`

Description:

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

Possible values:

`MSK_ON` Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- **max_num_warnings**

Corresponding constant:

MSK_IPAR_MAX_NUM_WARNINGS

Description:

Warning level. A higher value results in more warnings.

Possible Values:

Any number between 0 and +inf.

Default value:

10

- **mio_branch_dir**

Corresponding constant:

MSK_IPAR_MIO_BRANCH_DIR

Description:

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

Possible values:

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

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

MSK_BRANCH_DIR_FREE The mixed-integer optimizer decides which branch to choose.

Default value:

MSK_BRANCH_DIR_FREE

- **mio_branch_priorities_use**

Corresponding constant:

MSK_IPAR_MIO_BRANCH_PRIORITIES_USE

Description:

Controls whether branching priorities are used by the mixed-integer optimizer.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

• `mio_construct_sol`**Corresponding constant:**

MSK_IPAR_MIO_CONSTRUCT_SOL

Description:

If set to **MSK_ON** and all integer variables have been given a value for which a feasible mixed integer solution exists, then MOSEK generates an initial solution to the mixed integer problem by fixing all integer values and solving the remaining problem.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

• `mio_cont_sol`**Corresponding constant:**

MSK_IPAR_MIO_CONT_SOL

Description:

Controls the meaning of the interior-point and basic solutions in mixed integer problems.

Possible values:

MSK_MIO_CONT_SOL_ITG The reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. A solution is only reported in case the problem has a primal feasible solution.

MSK_MIO_CONT_SOL_NONE No interior-point or basic solution are reported when the mixed-integer optimizer is used.

MSK_MIO_CONT_SOL_ROOT The reported interior-point and basic solutions are a solution to the root node problem when mixed-integer optimizer is used.

MSK_MIO_CONT_SOL_ITG_REL In case the problem is primal feasible then the reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. If the problem is primal infeasible, then the solution to the root node problem is reported.

Default value:

MSK_MIO_CONT_SOL_NONE

- `mio_cut_level_root`

Corresponding constant:

MSK_IPAR_MIO_CUT_LEVEL_ROOT

Description:

Controls the cut level employed by the mixed-integer optimizer at the root node. A negative value means a default value determined by the mixed-integer optimizer is used. By adding the appropriate values from the following table the employed cut types can be controlled.

GUB cover	+2
Flow cover	+4
Lifting	+8
Plant location	+16
Disaggregation	+32
Knapsack cover	+64
Lattice	+128
Gomory	+256
Coefficient reduction	+512
GCD	+1024
Obj. integrality	+2048

Possible Values:

Any value.

Default value:

-1

- `mio_cut_level_tree`

Corresponding constant:

MSK_IPAR_MIO_CUT_LEVEL_TREE

Description:

Controls the cut level employed by the mixed-integer optimizer at the tree. See [MSK_IPAR_MIO_CUT_LEVEL_ROOT](#) for an explanation of the parameter values.

Possible Values:

Any value.

Default value:

-1

- `mio_feaspump_level`

Corresponding constant:

MSK_IPAR_MIO_FEASPUMP_LEVEL

Description:

Feasibility pump is a heuristic designed to compute an initial feasible solution. A value of 0 implies that the feasibility pump heuristic is not used. A value of -1 implies that the mixed-integer optimizer decides how the feasibility pump heuristic is used. A larger value than 1 implies that the feasibility pump is employed more aggressively. Normally a value beyond 3 is not worthwhile.

Possible Values:

Any number between -inf and 3.

Default value:

-1

- `mio_heuristic_level`

Corresponding constant:

`MSK_IPAR_MIO_HEURISTIC_LEVEL`

Description:

Controls the heuristic employed by the mixed-integer optimizer to locate an initial good integer feasible solution. A value of zero means the heuristic is not used at all. A larger value than 0 means that a gradually more sophisticated heuristic is used which is computationally more expensive. A negative value implies that the optimizer chooses the heuristic. Normally a value around 3 to 5 should be optimal.

Possible Values:

Any value.

Default value:

-1

- `mio_hotstart`

Corresponding constant:

`MSK_IPAR_MIO_HOTSTART`

Description:

Controls whether the integer optimizer is hot-started.

Possible values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

`MSK_ON`

- `mio_keep_basis`

Corresponding constant:

MSK_IPAR_MIO_KEEP_BASIS

Description:

Controls whether the integer presolve keeps bases in memory. This speeds on the solution process at cost of bigger memory consumption.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `mio_local_branch_number`

Corresponding constant:

MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER

Description:

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

Possible Values:

Any number between -inf and +inf.

Default value:

-1

- `mio_max_num_branches`

Corresponding constant:

MSK_IPAR_MIO_MAX_NUM_BRANCHES

Description:

Maximum number of branches allowed during the branch and bound search. A negative value means infinite.

Possible Values:

Any number between -inf and +inf.

Default value:

-1

See also:

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

- `mio_max_num_relaxs`

Corresponding constant:

MSK_IPAR_MIO_MAX_NUM_RELAXS

Description:

Maximum number of relaxations allowed during the branch and bound search. A negative value means infinite.

Possible Values:

Any number between -inf and +inf.

Default value:

-1

See also:

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

- `mio_max_num_solutions`

Corresponding constant:

MSK_IPAR_MIO_MAX_NUM_SOLUTIONS

Description:

The mixed-integer optimizer can be terminated after a certain number of different feasible solutions has been located. If this parameter has the value n and n is strictly positive, then the mixed-integer optimizer will be terminated when n feasible solutions have been located.

Possible Values:

Any number between -inf and +inf.

Default value:

-1

See also:

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

- `mio_mode`

Corresponding constant:

MSK_IPAR_MIO_MODE

Description:

Controls whether the optimizer includes the integer restrictions when solving a (mixed) integer optimization problem.

Possible values:

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

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

`MSK_MIO_MODE_SATISFIED` Integer restrictions should be satisfied.

Default value:

`MSK_MIO_MODE_SATISFIED`

- `mio_node_optimizer`

Corresponding constant:

`MSK_IPAR_MIO_NODE_OPTIMIZER`

Description:

Controls which optimizer is employed at the non-root nodes in the mixed-integer optimizer.

Possible values:

`MSK_OPTIMIZER_INTPNT` The interior-point optimizer is used.

`MSK_OPTIMIZER_CONCURRENT` The optimizer for nonconvex nonlinear problems.

`MSK_OPTIMIZER_MIXED_INT` The mixed-integer optimizer.

`MSK_OPTIMIZER_DUAL_SIMPLEX` The dual simplex optimizer is used.

`MSK_OPTIMIZER_FREE` The optimizer is chosen automatically.

`MSK_OPTIMIZER_PRIMAL_DUAL_SIMPLEX` The primal dual simplex optimizer is used.

`MSK_OPTIMIZER_CONIC` The optimizer for problems having conic constraints.

`MSK_OPTIMIZER_NONCONVEX` The optimizer for nonconvex nonlinear problems.

`MSK_OPTIMIZER_QCONE` For internal use only.

`MSK_OPTIMIZER_PRIMAL_SIMPLEX` The primal simplex optimizer is used.

`MSK_OPTIMIZER_FREE_SIMPLEX` One of the simplex optimizers is used.

Default value:

`MSK_OPTIMIZER_FREE`

- `mio_node_selection`

Corresponding constant:

`MSK_IPAR_MIO_NODE_SELECTION`

Description:

Controls the node selection strategy employed by the mixed-integer optimizer.

Possible values:

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

MSK_MIO_NODE_SELECTION_HYBRID The optimizer employs a hybrid strategy.

MSK_MIO_NODE_SELECTION_FREE The optimizer decides the node selection strategy.

MSK_MIO_NODE_SELECTION_WORST The optimizer employs a worst bound node selection strategy.

MSK_MIO_NODE_SELECTION_BEST The optimizer employs a best bound node selection strategy.

MSK_MIO_NODE_SELECTION_FIRST The optimizer employs a depth first node selection strategy.

Default value:

MSK_MIO_NODE_SELECTION_FREE

- `mio_optimizer_mode`

Corresponding constant:

MSK_IPAR_MIO_OPTIMIZER_MODE

Description:

An experimental feature.

Possible Values:

Any number between 0 and 1.

Default value:

0

- `mio_presolve_aggregate`

Corresponding constant:

MSK_IPAR_MIO_PREOLVEAggregate

Description:

Controls whether the presolve used by the mixed-integer optimizer tries to aggregate the constraints.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `mio_presolve_probing`

Corresponding constant:

MSK_IPAR_MIO_PREOLVEProbing

Description:

Controls whether the mixed-integer presolve performs probing. Probing can be very time consuming.

Possible values:

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

Default value:

MSK_ON

- `mio_presolve_use`

Corresponding constant:

MSK_IPAR_MIO_PRESOLVE_USE

Description:

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

Possible values:

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

Default value:

MSK_ON

- `mio_root_optimizer`

Corresponding constant:

MSK_IPAR_MIO_ROOT_OPTIMIZER

Description:

Controls which optimizer is employed at the root node in the mixed-integer optimizer.

Possible values:

MSK_OPTIMIZER_INTPNT The interior-point optimizer is used.
MSK_OPTIMIZER_CONCURRENT The optimizer for nonconvex nonlinear problems.
MSK_OPTIMIZER_MIXED_INT The mixed-integer optimizer.
MSK_OPTIMIZER_DUAL_SIMPLEX The dual simplex optimizer is used.
MSK_OPTIMIZER_FREE The optimizer is chosen automatically.
MSK_OPTIMIZER_PRIMAL_DUAL_SIMPLEX The primal dual simplex optimizer is used.
MSK_OPTIMIZER_CONIC The optimizer for problems having conic constraints.
MSK_OPTIMIZER_NONCONVEX The optimizer for nonconvex nonlinear problems.
MSK_OPTIMIZER_QCONE For internal use only.

MSK_OPTIMIZER_PRIMAL_SIMPLEX The primal simplex optimizer is used.

MSK_OPTIMIZER_FREE_SIMPLEX One of the simplex optimizers is used.

Default value:

MSK_OPTIMIZER_FREE

- `mio_strong_branch`

Corresponding constant:

MSK_IPAR_MIO_STRONG_BRANCH

Description:

The value specifies the depth from the root in which strong branching is used. A negative value means that the optimizer chooses a default value automatically.

Possible Values:

Any number between -inf and +inf.

Default value:

-1

- `nonconvex_max_iterations`

Corresponding constant:

MSK_IPAR_NONCONVEX_MAX_ITERATIONS

Description:

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

Possible Values:

Any number between 0 and +inf.

Default value:

100000

- `objective_sense`

Corresponding constant:

MSK_IPAR_OBJECTIVE_SENSE

Description:

If the objective sense for the task is undefined, then the value of this parameter is used as the default objective sense.

Possible values:

MSK_OBJECTIVE_SENSE_MINIMIZE The problem should be minimized.

MSK_OBJECTIVE_SENSE_UNDEFINED The objective sense is undefined.

MSK_OBJECTIVE_SENSE_MAXIMIZE The problem should be maximized.

Default value:

MSK_OBJECTIVE_SENSE_MINIMIZE

- opf_max_terms_per_line

Corresponding constant:

MSK_IPAR_OPF_MAX_TERMS_PER_LINE

Description:

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

Possible Values:

Any number between 0 and +inf.

Default value:

5

- opf_write_header

Corresponding constant:

MSK_IPAR_OPF_WRITE_HEADER

Description:

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

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- opf_write_hints

Corresponding constant:

MSK_IPAR_OPF_WRITE_HINTS

Description:

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

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- opf_write_parameters

Corresponding constant:

MSK_IPAR_OPF_WRITE_PARAMETERS

Description:

Write a parameter section in an OPF file.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- opf_write_problem

Corresponding constant:

MSK_IPAR_OPF_WRITE_PROBLEM

Description:

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

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- opf_write_sol_bas

Corresponding constant:

MSK_IPAR_OPF_WRITE_SOL_BAS

Description:

If **MSK_IPAR_OPF_WRITE_SOLUTIONS** is **MSK_ON** and a basic solution is defined, include the basic solution in OPF files.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- opf_write_sol_itg

Corresponding constant:

MSK_IPAR_OPF_WRITE_SOL_ITG

Description:

If **MSK_IPAR_OPF_WRITE_SOLUTIONS** is **MSK_ON** and an integer solution is defined, write the integer solution in OPF files.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `opf_write_sol_itr`

Corresponding constant:

MSK_IPAR_OPF_WRITE_SOL_ITR

Description:

If **MSK_IPAR_OPF_WRITE_SOLUTIONS** is **MSK_ON** and an interior solution is defined, write the interior solution in OPF files.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `opf_write_solutions`

Corresponding constant:

MSK_IPAR_OPF_WRITE_SOLUTIONS

Description:

Enable inclusion of solutions in the OPF files.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- `optimizer`

Corresponding constant:

MSK_IPAR_OPTIMIZER

Description:

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

Possible values:

MSK_OPTIMIZER_INTPNT The interior-point optimizer is used.
 MSK_OPTIMIZER_CONCURRENT The optimizer for nonconvex nonlinear problems.
 MSK_OPTIMIZER_MIXED_INT The mixed-integer optimizer.
 MSK_OPTIMIZER_DUAL_SIMPLEX The dual simplex optimizer is used.
 MSK_OPTIMIZER_FREE The optimizer is chosen automatically.
 MSK_OPTIMIZER_PRIMAL_DUAL_SIMPLEX The primal dual simplex optimizer is used.
 MSK_OPTIMIZER_CONIC The optimizer for problems having conic constraints.
 MSK_OPTIMIZER_NONCONVEX The optimizer for nonconvex nonlinear problems.
 MSK_OPTIMIZER_QCONE For internal use only.
 MSK_OPTIMIZER_PRIMAL_SIMPLEX The primal simplex optimizer is used.
 MSK_OPTIMIZER_FREE_SIMPLEX One of the simplex optimizers is used.

Default value:

MSK_OPTIMIZER_FREE

- param_read_case_name

Corresponding constant:

MSK_IPAR_PARAM_READ_CASE_NAME

Description:

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

Possible values:

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

Default value:

MSK_ON

- param_read_ign_error

Corresponding constant:

MSK_IPAR_PARAM_READ_IGN_ERROR

Description:

If turned on, then errors in parameter settings is ignored.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- `presolve_elim_fill`

Corresponding constant:

MSK_IPAR_PREOLVE_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 $+\infty$.

Default value:

1

- `presolve_eliminator_max_num_tries`

Corresponding constant:

MSK_IPAR_PREOLVE_ELIMINATOR_MAX_NUM_TRIES

Description:

Control the maximum number of times the eliminator is tried.

Possible Values:

A negative value implies MOSEK decides maximum number of times.

Default value:

-1

- `presolve_eliminator_use`

Corresponding constant:

MSK_IPAR_PREOLVE_ELIMINATOR_USE

Description:

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

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `presolve_level`

Corresponding constant:

`MSK_IPAR_PRESOLVE_LEVEL`

Description:

Currently not used.

Possible Values:

Any number between -inf and +inf.

Default value:

-1

- `presolve_lindep_use`

Corresponding constant:

`MSK_IPAR_PRESOLVE_LINDEP_USE`

Description:

Controls whether the linear constraints are checked for linear dependencies.

Possible values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

`MSK_ON`

- `presolve_lindep_work_lim`

Corresponding constant:

`MSK_IPAR_PRESOLVE_LINDEP_WORK_LIM`

Description:

Is used to limit the amount of work that can be done to locate linear dependencies. In general the higher value this parameter is given the less work can be used. However, a value of 0 means no limit on the amount of work that can be used.

Possible Values:

Any number between 0 and +inf.

Default value:

1

- `presolve_use`

Corresponding constant:

`MSK_IPAR_PRESOLVE_USE`

Description:

Controls whether the presolve is applied to a problem before it is optimized.

Possible values:

MSK_PRESOLVE_MODE_ON The problem is presolved before it is optimized.

MSK_PRESOLVE_MODE_OFF The problem is not presolved before it is optimized.

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

Default value:

MSK_PRESOLVE_MODE_FREE

- `qo_separable_reformulation`

Corresponding constant:

MSK_IPAR_QO_SEPARABLE_REFORMULATION

Description:

Determine if Quadratic programming problems should be reformulated to separable form.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- `read_add_anz`

Corresponding constant:

MSK_IPAR_READ_ADD_ANZ

Description:

Additional number of non-zeros in A that is made room for in the problem.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

0

- `read_add_con`

Corresponding constant:

MSK_IPAR_READ_ADD_CON

Description:

Additional number of constraints that is made room for in the problem.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

0

- read_add_cone

Corresponding constant:

MSK_IPAR_READ_ADD_CONE

Description:

Additional number of conic constraints that is made room for in the problem.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

0

- read_add_qnz

Corresponding constant:

MSK_IPAR_READ_ADD_QNZ

Description:

Additional number of non-zeros in the Q matrices that is made room for in the problem.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

0

- read_add_var

Corresponding constant:

MSK_IPAR_READ_ADD_VAR

Description:

Additional number of variables that is made room for in the problem.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

0

- read_anz

Corresponding constant:

MSK_IPAR_READ_ANZ

Description:

Expected maximum number of A non-zeros to be read. The option is used only by fast MPS and LP file readers.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

100000

• read_con

Corresponding constant:

MSK_IPAR_READ_CON

Description:

Expected maximum number of constraints to be read. The option is only used by fast MPS and LP file readers.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

10000

• read_cone

Corresponding constant:

MSK_IPAR_READ_CONE

Description:

Expected maximum number of conic constraints to be read. The option is used only by fast MPS and LP file readers.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

2500

• read_data_compressed

Corresponding constant:

MSK_IPAR_READ_DATA_COMPRESSED

Description:

If this option is turned on, it is assumed that the data file is compressed.

Possible values:

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

Default value:

MSK_OFF

- read_data_format

Corresponding constant:

MSK_IPAR_READ_DATA_FORMAT

Description:

Format of the data file to be read.

Possible values:

MSK_DATA_FORMAT_XML The data file is an XML formatted file.
 MSK_DATA_FORMAT_FREE_MPS The data data a free MPS formatted file.
 MSK_DATA_FORMAT_EXTENSION The file extension is used to determine the data file format.
 MSK_DATA_FORMAT_MPS The data file is MPS formatted.
 MSK_DATA_FORMAT_LP The data file is LP formatted.
 MSK_DATA_FORMAT_MBT The data file is a MOSEK binary task file.
 MSK_DATA_FORMAT_OP The data file is an optimization problem formatted file.

Default value:

MSK_DATA_FORMAT_EXTENSION

- read_keep_free_con

Corresponding constant:

MSK_IPAR_READ_KEEP_FREE_CON

Description:

Controls whether the free constraints are included in the problem.

Possible values:

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

Default value:

MSK_OFF

- read_lp_drop_new_vars_in_bou

Corresponding constant:

MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU

Description:

If this option is turned on, MOSEK will drop variables that are defined for the first time in the bounds section.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

• read_lp_quoted_names

Corresponding constant:

MSK_IPAR_READ_LP_QUOTED_NAMES

Description:

If a name is in quotes when reading an LP file, the quotes will be removed.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

• read_mps_format

Corresponding constant:

MSK_IPAR_READ_MPS_FORMAT

Description:

Controls how strictly the MPS file reader interprets the MPS format.

Possible values:

MSK_MPS_FORMAT_STRICT It is assumed that the input file satisfies the MPS format strictly.

MSK_MPS_FORMAT_RELAXED It is assumed that the input file satisfies a slightly relaxed version of the MPS format.

MSK_MPS_FORMAT_FREE It is assumed that the input file satisfies the free MPS format. This implies that spaces are not allowed in names. Otherwise the format is free.

Default value:

MSK_MPS_FORMAT_RELAXED

- `read_mps_keep_int`

Corresponding constant:

`MSK_IPAR_READ_MPS_KEEP_INT`

Description:

Controls whether MOSEK should keep the integer restrictions on the variables while reading the MPS file.

Possible values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

`MSK_ON`

- `read_mps_obj_sense`

Corresponding constant:

`MSK_IPAR_READ_MPS_OBJ_SENSE`

Description:

If turned on, the MPS reader uses the objective sense section. Otherwise the MPS reader ignores it.

Possible values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

`MSK_ON`

- `read_mps_quoted_names`

Corresponding constant:

`MSK_IPAR_READ_MPS_QUOTED_NAMES`

Description:

If a name is in quotes when reading an MPS file, then the quotes will be removed.

Possible values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

`MSK_ON`

- `read_mps_relax`

Corresponding constant:

MSK_IPAR_READ_MPS_RELAX

Description:

If this option is turned on, then mixed integer constraints are ignored when a problem is read.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- read_mps_width

Corresponding constant:

MSK_IPAR_READ_MPS_WIDTH

Description:

Controls the maximal number of characters allowed in one line of the MPS file.

Possible Values:

Any positive number greater than 80.

Default value:

1024

- read_q_mode

Corresponding constant:

MSK_IPAR_READ_Q_MODE

Description:

Controls how the Q matrices are read from the MPS file.

Possible values:

MSK_Q_READ_ADD All elements in a Q matrix are assumed to belong to the lower triangular part. Duplicate elements in a Q matrix are added together.

MSK_Q_READ_DROP_LOWER All elements in the strict lower triangular part of the Q matrices are dropped.

MSK_Q_READ_DROP_UPPER All elements in the strict upper triangular part of the Q matrices are dropped.

Default value:

MSK_Q_READ_ADD

- read_qnz

Corresponding constant:

MSK_IPAR_READ_QNZ

Description:

Expected maximum number of Q non-zeros to be read. The option is used only by MPS and LP file readers.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

20000

- read_task_ignore_param

Corresponding constant:

MSK_IPAR_READ_TASK_IGNORE_PARAM

Description:

Controls whether MOSEK should ignore the parameter setting defined in the task file and use the default parameter setting instead.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- read_var

Corresponding constant:

MSK_IPAR_READ_VAR

Description:

Expected maximum number of variable to be read. The option is used only by MPS and LP file readers.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

10000

- sensitivity_optimizer

Corresponding constant:

MSK_IPAR_SENSITIVITY_OPTIMIZER

Description:

Controls which optimizer is used for optimal partition sensitivity analysis.

Possible values:

MSK_OPTIMIZER_INTPNT The interior-point optimizer is used.
 MSK_OPTIMIZER_CONCURRENT The optimizer for nonconvex nonlinear problems.
 MSK_OPTIMIZER_MIXED_INT The mixed-integer optimizer.
 MSK_OPTIMIZER_DUAL_SIMPLEX The dual simplex optimizer is used.
 MSK_OPTIMIZER_FREE The optimizer is chosen automatically.
 MSK_OPTIMIZER_PRIMAL_DUAL_SIMPLEX The primal dual simplex optimizer is used.
 MSK_OPTIMIZER_CONIC The optimizer for problems having conic constraints.
 MSK_OPTIMIZER_NONCONVEX The optimizer for nonconvex nonlinear problems.
 MSK_OPTIMIZER_QCONE For internal use only.
 MSK_OPTIMIZER_PRIMAL_SIMPLEX The primal simplex optimizer is used.
 MSK_OPTIMIZER_FREE_SIMPLEX One of the simplex optimizers is used.

Default value:

MSK_OPTIMIZER_FREE_SIMPLEX

- `sensitivity_type`

Corresponding constant:

MSK_IPAR_SENSITIVITY_TYPE

Description:

Controls which type of sensitivity analysis is to be performed.

Possible values:

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

Default value:

MSK_SENSITIVITY_TYPE_BASIS

- `sim_basis_factor_use`

Corresponding constant:

MSK_IPAR_SIM_BASIS_FACTOR_USE

Description:

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

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `sim_degen`

Corresponding constant:

MSK_IPAR_SIM_DEGEN

Description:

Controls how aggressively degeneration is handled.

Possible values:

MSK_SIM_DEGEN_NONE The simplex optimizer should use no degeneration strategy.

MSK_SIM_DEGEN_MODERATE The simplex optimizer should use a moderate degeneration strategy.

MSK_SIM_DEGEN_MINIMUM The simplex optimizer should use a minimum degeneration strategy.

MSK_SIM_DEGEN_AGGRESSIVE The simplex optimizer should use an aggressive degeneration strategy.

MSK_SIM_DEGEN_FREE The simplex optimizer chooses the degeneration strategy.

Default value:

MSK_SIM_DEGEN_FREE

- `sim_dual_crash`

Corresponding constant:

MSK_IPAR_SIM_DUAL_CRASH

Description:

Controls whether crashing is performed in the dual simplex optimizer.

In general if a basis consists of more than (100-this parameter value)% fixed variables, then a crash will be performed.

Possible Values:

Any number between 0 and +inf.

Default value:

90

- `sim_dual_phaseone_method`

Corresponding constant:

MSK_IPAR_SIM_DUAL_PHASEONE_METHOD

Description:

An experimental feature.

Possible Values:

Any number between 0 and 10.

Default value:

0

- `sim_dual_restrict_selection`

Corresponding constant:

MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION

Description:

The dual simplex optimizer can use a so-called restricted selection/pricing strategy to choose the outgoing variable. Hence, if restricted selection is applied, then the dual simplex optimizer first chooses a subset of all the potential outgoing variables. Next, for some time it will choose the outgoing variable only among the subset. From time to time the subset is redefined.

A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.

Possible Values:

Any number between 0 and 100.

Default value:

50

- `sim_dual_selection`

Corresponding constant:

MSK_IPAR_SIM_DUAL_SELECTION

Description:

Controls the choice of the incoming variable, known as the selection strategy, in the dual simplex optimizer.

Possible values:

MSK_SIM_SELECTION_FULL The optimizer uses full pricing.

MSK_SIM_SELECTION_PARTIAL The optimizer uses a partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.

MSK_SIM_SELECTION_FREE The optimizer chooses the pricing strategy.

MSK_SIM_SELECTION_ASE The optimizer uses approximate steepest-edge pricing.

MSK_SIM_SELECTION_DEVEX The optimizer uses devex steepest-edge pricing (or if it is not available an approximate steep-edge selection).

MSK_SIM_SELECTION_SE The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

Default value:

MSK_SIM_SELECTION_FREE

- `sim_exploit_dupvec`

Corresponding constant:

MSK_IPAR_SIM_EXPLOIT_DUPVEC

Description:

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

Possible values:

MSK_SIM_EXPLOIT_DUPVEC_ON Allow the simplex optimizer to exploit duplicated columns.

MSK_SIM_EXPLOIT_DUPVEC_OFF Disallow the simplex optimizer to exploit duplicated columns.

MSK_SIM_EXPLOIT_DUPVEC_FREE The simplex optimizer can choose freely.

Default value:

MSK_SIM_EXPLOIT_DUPVEC_OFF

- `sim_hotstart`

Corresponding constant:

MSK_IPAR_SIM_HOTSTART

Description:

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

Possible values:

MSK_SIM_HOTSTART_NONE The simplex optimizer performs a coldstart.

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

MSK_SIM_HOTSTART_FREE The simplex optimizer chooses the hot-start type.

Default value:

MSK_SIM_HOTSTART_FREE

- `sim_hotstart_lu`

Corresponding constant:

MSK_IPAR_SIM_HOTSTART_LU

Description:

Determines if the simplex optimizer should exploit the initial factorization.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

• `sim_integer`**Corresponding constant:**

MSK_IPAR_SIM_INTEGER

Description:

An experimental feature.

Possible Values:

Any number between 0 and 10.

Default value:

0

• `sim_max_iterations`**Corresponding constant:**

MSK_IPAR_SIM_MAX_ITERATIONS

Description:

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

Possible Values:Any number between 0 and $+\infty$.**Default value:**

10000000

• `sim_max_num_setbacks`**Corresponding constant:**

MSK_IPAR_SIM_MAX_NUM_SETBACKS

Description:

Controls how many set-backs are allowed within a simplex optimizer. A set-back is an event where the optimizer moves in the wrong direction. This is impossible in theory but may happen due to numerical problems.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

250

- `sim_network_detect`

Corresponding constant:

`MSK_IPAR_SIM_NETWORK_DETECT`

Description:

The simplex optimizer is capable of exploiting a network flow component in a problem. However it is only worthwhile to exploit the network flow component if it is sufficiently large. This parameter controls how large the network component has to be in “relative” terms before it is exploited. For instance a value of 20 means at least 20% of the model should be a network before it is exploited. If this value is larger than 100 the network flow component is never detected or exploited.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

101

- `sim_network_detect_hotstart`

Corresponding constant:

`MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART`

Description:

This parameter controls how large the network component in “relative” terms has to be before it is exploited in a simplex hot-start. The network component should be equal or larger than

$\max(\text{MSK_IPAR_SIM_NETWORK_DETECT}, \text{MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART})$

before it is exploited. If this value is larger than 100 the network flow component is never detected or exploited.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

100

- `sim_network_detect_method`

Corresponding constant:

`MSK_IPAR_SIM_NETWORK_DETECT_METHOD`

Description:

Controls which type of detection method the network extraction should use.

Possible values:

MSK_NETWORK_DETECT_SIMPLE The network detection should use a very simple heuristic.

MSK_NETWORK_DETECT_ADVANCED The network detection should use a more advanced heuristic.

MSK_NETWORK_DETECT_FREE The network detection is free.

Default value:

MSK_NETWORK_DETECT_FREE

- `sim_non_singular`

Corresponding constant:

MSK_IPAR_SIM_NON_SINGULAR

Description:

Controls if the simplex optimizer ensures a non-singular basis, if possible.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `sim_primal_crash`

Corresponding constant:

MSK_IPAR_SIM_PRIMAL_CRASH

Description:

Controls whether crashing is performed in the primal simplex optimizer.

In general, if a basis consists of more than (100-this parameter value)% fixed variables, then a crash will be performed.

Possible Values:

Any nonnegative integer value.

Default value:

90

- `sim_primal_phaseone_method`

Corresponding constant:

MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD

Description:

An experimental feature.

Possible Values:

Any number between 0 and 10.

Default value:

0

- `sim_primal_restrict_selection`

Corresponding constant:

`MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION`

Description:

The primal simplex optimizer can use a so-called restricted selection/pricing strategy to choose the outgoing variable. Hence, if restricted selection is applied, then the primal simplex optimizer first chooses a subset of all the potential incoming variables. Next, for some time it will choose the incoming variable only among the subset. From time to time the subset is redefined.

A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.

Possible Values:

Any number between 0 and 100.

Default value:

50

- `sim_primal_selection`

Corresponding constant:

`MSK_IPAR_SIM_PRIMAL_SELECTION`

Description:

Controls the choice of the incoming variable, known as the selection strategy, in the primal simplex optimizer.

Possible values:

`MSK_SIM_SELECTION_FULL` The optimizer uses full pricing.

`MSK_SIM_SELECTION_PARTIAL` The optimizer uses a partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.

`MSK_SIM_SELECTION_FREE` The optimizer chooses the pricing strategy.

`MSK_SIM_SELECTION_ASE` The optimizer uses approximate steepest-edge pricing.

MSK_SIM_SELECTION_DEVEX The optimizer uses devex steepest-edge pricing (or if it is not available an approximate steep-edge selection).

MSK_SIM_SELECTION_SE The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

Default value:

MSK_SIM_SELECTION_FREE

- `sim_refactor_freq`

Corresponding constant:

MSK_IPAR_SIM_REFACTOR_FREQ

Description:

Controls how frequent the basis is refactorized. The value 0 means that the optimizer determines the best point of refactorization.

It is strongly recommended NOT to change this parameter.

Possible Values:

Any number between 0 and +inf.

Default value:

0

- `sim_reformulation`

Corresponding constant:

MSK_IPAR_SIM_REFORMULATION

Description:

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

Possible values:

MSK_SIM_REFORMULATION_ON Allow the simplex optimizer to reformulate the problem.

MSK_SIM_REFORMULATION_AGGRESSIVE The simplex optimizer should use an aggressive reformulation strategy.

MSK_SIM_REFORMULATION_OFF Disallow the simplex optimizer to reformulate the problem.

MSK_SIM_REFORMULATION_FREE The simplex optimizer can choose freely.

Default value:

MSK_SIM_REFORMULATION_OFF

- `sim_save_lu`

Corresponding constant:

MSK_IPAR_SIM_SAVE_LU

Description:

Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.

Possible values:

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

Default value:

MSK_OFF

- `sim_scaling`

Corresponding constant:

MSK_IPAR_SIM_SCALING

Description:

Controls how much effort is used in scaling the problem before a simplex optimizer is used.

Possible values:

MSK_SCALING_NONE No scaling is performed.
MSK_SCALING_MODERATE A conservative scaling is performed.
MSK_SCALING_AGGRESSIVE A very aggressive scaling is performed.
MSK_SCALING_FREE The optimizer chooses the scaling heuristic.

Default value:

MSK_SCALING_FREE

- `sim_scaling_method`

Corresponding constant:

MSK_IPAR_SIM_SCALING_METHOD

Description:

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

Possible values:

MSK_SCALING_METHOD_POW2 Scales only with power of 2 leaving the mantissa untouched.
MSK_SCALING_METHOD_FREE The optimizer chooses the scaling heuristic.

Default value:

MSK_SCALING_METHOD_POW2

- `sim_solve_form`

Corresponding constant:

MSK_IPAR_SIM_SOLVE_FORM

Description:

Controls whether the primal or the dual problem is solved by the primal-/dual-simplex optimizer.

Possible values:

MSK_SOLVE_PRIMAL The optimizer should solve the primal problem.

MSK_SOLVE_DUAL The optimizer should solve the dual problem.

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

Default value:

MSK_SOLVE_FREE

- `sim_stability_priority`

Corresponding constant:

MSK_IPAR_SIM_STABILITY_PRIORITY

Description:

Controls how high priority the numerical stability should be given.

Possible Values:

Any number between 0 and 100.

Default value:

50

- `sim_switch_optimizer`

Corresponding constant:

MSK_IPAR_SIM_SWITCH_OPTIMIZER

Description:

The simplex optimizer sometimes chooses to solve the dual problem instead of the primal problem. This implies that if you have chosen to use the dual simplex optimizer and the problem is dualized, then it actually makes sense to use the primal simplex optimizer instead. If this parameter is on and the problem is dualized and furthermore the simplex optimizer is chosen to be the primal (dual) one, then it is switched to the dual (primal).

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

• `sol_filter_keep_basic`**Corresponding constant:**

MSK_IPAR_SOL_FILTER_KEEP_BASIC

Description:

If turned on, then basic and super basic constraints and variables are written to the solution file independent of the filter setting.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

• `sol_filter_keep_ranged`**Corresponding constant:**

MSK_IPAR_SOL_FILTER_KEEP_RANGED

Description:

If turned on, then ranged constraints and variables are written to the solution file independent of the filter setting.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

• `sol_quoted_names`**Corresponding constant:**

MSK_IPAR_SOL_QUOTED_NAMES

Description:

If this options is turned on, then MOSEK will quote names that contains blanks while writing the solution file. Moreover when reading leading and trailing quotes will be stripped of.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- `sol_read_name_width`

Corresponding constant:

MSK_IPAR_SOL_READ_NAME_WIDTH

Description:

When a solution is read by MOSEK and some constraint, variable or cone names contain blanks, then a maximum name width must be specified. A negative value implies that no name contain blanks.

Possible Values:

Any number between -inf and +inf.

Default value:

-1

- `sol_read_width`

Corresponding constant:

MSK_IPAR_SOL_READ_WIDTH

Description:

Controls the maximal acceptable width of line in the solutions when read by MOSEK.

Possible Values:

Any positive number greater than 80.

Default value:

1024

- `solution_callback`

Corresponding constant:

MSK_IPAR_SOLUTION_CALLBACK

Description:

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

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- `timing_level`

Corresponding constant:

`MSK_IPAR_TIMING_LEVEL`

Description:

Controls the amount of timing performed inside MOSEK.

Possible Values:

Any integer greater or equal to 0.

Default value:

1

- `warning_level`

Corresponding constant:

`MSK_IPAR_WARNING_LEVEL`

Description:

Warning level.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

1

- `write_bas_constraints`

Corresponding constant:

`MSK_IPAR_WRITE_BAS_CONSTRAINTS`

Description:

Controls whether the constraint section is written to the basic solution file.

Possible values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

`MSK_ON`

- `write_bas_head`

Corresponding constant:

`MSK_IPAR_WRITE_BAS_HEAD`

Description:

Controls whether the header section is written to the basic solution file.

Possible values:

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

Default value:

MSK_ON

- write_bas_variables

Corresponding constant:

MSK_IPAR_WRITE_BAS_VARIABLES

Description:

Controls whether the variables section is written to the basic solution file.

Possible values:

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

Default value:

MSK_ON

- write_data_compressed

Corresponding constant:

MSK_IPAR_WRITE_DATA_COMPRESSED

Description:

Controls whether the data file is compressed while it is written. 0 means no compression while higher values mean more compression.

Possible Values:

Any number between 0 and +inf.

Default value:

0

- write_data_format

Corresponding constant:

MSK_IPAR_WRITE_DATA_FORMAT

Description:

Controls the file format when writing task data to a file.

Possible values:

MSK_DATA_FORMAT_XML The data file is an XML formatted file.

MSK_DATA_FORMAT_FREE_MPS The data data a free MPS formatted file.

MSK_DATA_FORMAT_EXTENSION The file extension is used to determine the data file format.

MSK_DATA_FORMAT_MPS The data file is MPS formatted.

MSK_DATA_FORMAT_LP The data file is LP formatted.

MSK_DATA_FORMAT_MBT The data file is a MOSEK binary task file.

MSK_DATA_FORMAT_OP The data file is an optimization problem formatted file.

Default value:

MSK_DATA_FORMAT_EXTENSION

- write_data_param

Corresponding constant:

MSK_IPAR_WRITE_DATA_PARAM

Description:

If this option is turned on the parameter settings are written to the data file as parameters.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- write_free_con

Corresponding constant:

MSK_IPAR_WRITE_FREE_CON

Description:

Controls whether the free constraints are written to the data file.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- write_generic_names

Corresponding constant:

MSK_IPAR_WRITE_GENERIC_NAMES

Description:

Controls whether the generic names or user-defined names are used in the data file.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- `write_generic_names_io`

Corresponding constant:

MSK_IPAR_WRITE_GENERIC_NAMES_IO

Description:

Index origin used in generic names.

Possible Values:

Any number between 0 and +inf.

Default value:

1

- `write_int_constraints`

Corresponding constant:

MSK_IPAR_WRITE_INT_CONSTRAINTS

Description:

Controls whether the constraint section is written to the integer solution file.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `write_int_head`

Corresponding constant:

MSK_IPAR_WRITE_INT_HEAD

Description:

Controls whether the header section is written to the integer solution file.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `write_int_variables`

Corresponding constant:

MSK_IPAR_WRITE_INT_VARIABLES

Description:

Controls whether the variables section is written to the integer solution file.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `write_lp_line_width`

Corresponding constant:

MSK_IPAR_WRITE_LP_LINE_WIDTH

Description:

Maximum width of line in an LP file written by MOSEK.

Possible Values:

Any positive number.

Default value:

80

- `write_lp_quoted_names`

Corresponding constant:

MSK_IPAR_WRITE_LP_QUOTED_NAMES

Description:

If this option is turned on, then MOSEK will quote invalid LP names when writing an LP file.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `write_lp_strict_format`

Corresponding constant:

`MSK_IPAR_WRITE_LP_STRICT_FORMAT`

Description:

Controls whether LP output files satisfy the LP format strictly.

Possible values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

`MSK_OFF`

- `write_lp_terms_per_line`

Corresponding constant:

`MSK_IPAR_WRITE_LP_TERMS_PER_LINE`

Description:

Maximum number of terms on a single line in an LP file written by MOSEK. 0 means unlimited.

Possible Values:

Any number between 0 and $+\infty$.

Default value:

10

- `write_mps_int`

Corresponding constant:

`MSK_IPAR_WRITE_MPS_INT`

Description:

Controls if marker records are written to the MPS file to indicate whether variables are integer restricted.

Possible values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

`MSK_ON`

- `write_mps_obj_sense`

Corresponding constant:

MSK_IPAR_WRITE_MPS_OBJ_SENSE

Description:

If turned off, the objective sense section is not written to the MPS file.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `write_mps_quoted_names`

Corresponding constant:

MSK_IPAR_WRITE_MPS_QUOTED_NAMES

Description:

If a name contains spaces (blanks) when writing an MPS file, then the quotes will be removed.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- `write_mps_strict`

Corresponding constant:

MSK_IPAR_WRITE_MPS_STRICT

Description:

Controls whether the written MPS file satisfies the MPS format strictly or not.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_OFF

- `write_precision`

Corresponding constant:

MSK_IPAR_WRITE_PRECISION

Description:

Controls the precision with which `double` numbers are printed in the MPS data file. In general it is not worthwhile to use a value higher than 15.

Possible Values:

Any number between 0 and +inf.

Default value:

8

- `write_sol_constraints`

Corresponding constant:

`MSK_IPAR_WRITE_SOL_CONSTRAINTS`

Description:

Controls whether the constraint section is written to the solution file.

Possible values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

`MSK_ON`

- `write_sol_head`

Corresponding constant:

`MSK_IPAR_WRITE_SOL_HEAD`

Description:

Controls whether the header section is written to the solution file.

Possible values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

`MSK_ON`

- `write_sol_variables`

Corresponding constant:

`MSK_IPAR_WRITE_SOL_VARIABLES`

Description:

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

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- write_task_inc_sol

Corresponding constant:

MSK_IPAR_WRITE_TASK_INC_SOL

Description:

Controls whether the solutions are stored in the task file too.

Possible values:

MSK_ON Switch the option on.

MSK_OFF Switch the option off.

Default value:

MSK_ON

- write_xml_mode

Corresponding constant:

MSK_IPAR_WRITE_XML_MODE

Description:

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

Possible values:

MSK_WRITE_XML_MODE_COL Write in column order.

MSK_WRITE_XML_MODE_ROW Write in row order.

Default value:

MSK_WRITE_XML_MODE_ROW

E.4 String parameter types

- **MSK_SPAR_BAS_SOL_FILE_NAME** 378
Name of the bas solution file.
- **MSK_SPAR_DATA_FILE_NAME** 378
Data are read and written to this file.
- **MSK_SPAR_DEBUG_FILE_NAME** 378
MOSEK debug file.

- **MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL** 379
Feasibility repair name violation name.
- **MSK_SPAR_INT_SOL_FILE_NAME** 379
Name of the int solution file.
- **MSK_SPAR_ITR_SOL_FILE_NAME** 379
Name of the itr solution file.
- **MSK_SPAR_PARAM_COMMENT_SIGN** 379
Solution file comment character.
- **MSK_SPAR_PARAM_READ_FILE_NAME** 380
Modifications to the parameter database is read from this file.
- **MSK_SPAR_PARAM_WRITE_FILE_NAME** 380
The parameter database is written to this file.
- **MSK_SPAR_READ_MPS_BOU_NAME** 380
Name of the BOUNDS vector used. An empty name means that the first BOUNDS vector is used.
- **MSK_SPAR_READ_MPS_OBJ_NAME** 380
Objective name in the MPS file.
- **MSK_SPAR_READ_MPS_RAN_NAME** 381
Name of the RANGE vector used. An empty name means that the first RANGE vector is used.
- **MSK_SPAR_READ_MPS_RHS_NAME** 381
Name of the RHS used. An empty name means that the first RHS vector is used.
- **MSK_SPAR_SOL_FILTER_XC_LOW** 381
Solution file filter.
- **MSK_SPAR_SOL_FILTER_XC_UPR** 382
Solution file filter.
- **MSK_SPAR_SOL_FILTER_XX_LOW** 382
Solution file filter.
- **MSK_SPAR_SOL_FILTER_XX_UPR** 382
Solution file filter.
- **MSK_SPAR_STAT_FILE_NAME** 383
Statistics file name.

- **MSK_SPAR_STAT_KEY** 383
Key used when writing the summary file.
- **MSK_SPAR_STAT_NAME** 383
Name used when writing the statistics file.
- **MSK_SPAR_WRITE_LP_GEN_VAR_NAME** 383
Added variable names in the LP files.
- **bas_sol_file_name**

Corresponding constant:
MSK_SPAR_BAS_SOL_FILE_NAME
Description:
Name of the **bas** solution file.
Possible Values:
Any valid file name.
Default value:
""
- **data_file_name**

Corresponding constant:
MSK_SPAR_DATA_FILE_NAME
Description:
Data are read and written to this file.
Possible Values:
Any valid file name.
Default value:
""
- **debug_file_name**

Corresponding constant:
MSK_SPAR_DEBUG_FILE_NAME
Description:
MOSEK debug file.
Possible Values:
Any valid file name.
Default value:
""

- `feasrepair_name_wsumviol`

Corresponding constant:

`MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL`

Description:

The constraint and variable associated with the total weighted sum of violations are each given the name of this parameter postfixed with `CON` and `VAR` respectively.

Possible Values:

Any valid string.

Default value:

`"WSUMVIOL"`

- `int_sol_file_name`

Corresponding constant:

`MSK_SPAR_INT_SOL_FILE_NAME`

Description:

Name of the `int` solution file.

Possible Values:

Any valid file name.

Default value:

`""`

- `itr_sol_file_name`

Corresponding constant:

`MSK_SPAR_ITR_SOL_FILE_NAME`

Description:

Name of the `itr` solution file.

Possible Values:

Any valid file name.

Default value:

`""`

- `param_comment_sign`

Corresponding constant:

`MSK_SPAR_PARAM_COMMENT_SIGN`

Description:

Only the first character in this string is used. It is considered as a start of comment sign in the MOSEK parameter file. Spaces are ignored in the string.

Possible Values:

Any valid string.

Default value:

"%%"

- `param_read_file_name`

Corresponding constant:

`MSK_SPAR_PARAM_READ_FILE_NAME`

Description:

Modifications to the parameter database is read from this file.

Possible Values:

Any valid file name.

Default value:

""

- `param_write_file_name`

Corresponding constant:

`MSK_SPAR_PARAM_WRITE_FILE_NAME`

Description:

The parameter database is written to this file.

Possible Values:

Any valid file name.

Default value:

""

- `read_mps_bou_name`

Corresponding constant:

`MSK_SPAR_READ_MPS_BOU_NAME`

Description:

Name of the BOUNDS vector used. An empty name means that the first BOUNDS vector is used.

Possible Values:

Any valid MPS name.

Default value:

""

- `read_mps_obj_name`

Corresponding constant:

MSK_SPAR_READ_MPS_OBJ_NAME

Description:

Name of the free constraint used as objective function. An empty name means that the first constraint is used as objective function.

Possible Values:

Any valid MPS name.

Default value:

""

- `read_mps_ran_name`

Corresponding constant:

MSK_SPAR_READ_MPS_RAN_NAME

Description:

Name of the RANGE vector used. An empty name means that the first RANGE vector is used.

Possible Values:

Any valid MPS name.

Default value:

""

- `read_mps_rhs_name`

Corresponding constant:

MSK_SPAR_READ_MPS_RHS_NAME

Description:

Name of the RHS used. An empty name means that the first RHS vector is used.

Possible Values:

Any valid MPS name.

Default value:

""

- `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 $xc[i] \geq blc[i] + 0.5$ should be listed. An empty filter means that no filter is applied.

Possible Values:

Any valid filter.

Default value:

""

- `sol_filter_xc_upr`

Corresponding constant:

MSK_SPAR_SOL_FILTER_XC_UPR

Description:

A filter used to determine which constraints should be listed in the solution file.

A value of "0.5" means that all constraints having $xc[i] < 0.5$ should be listed, whereas "-0.5" means all constraints having $xc[i] \leq buc[i] - 0.5$ should be listed.

An empty filter means that no filter is applied.

Possible Values:

Any valid filter.

Default value:

""

- `sol_filter_xx_low`

Corresponding constant:

MSK_SPAR_SOL_FILTER_XX_LOW

Description:

A filter used to determine which variables should be listed in the solution file.

A value of "0.5" means that all constraints having $xx[j] \geq 0.5$ should be listed, whereas "+0.5" means that all constraints having $xx[j] \geq blx[j] + 0.5$ should be listed. An empty filter means no filter is applied.

Possible Values:

Any valid filter..

Default value:

""

- `sol_filter_xx_upr`

Corresponding constant:

MSK_SPAR_SOL_FILTER_XX_UPR

Description:

A filter used to determine which variables should be listed in the solution file.

A value of "0.5" means that all constraints having $xx[j] < 0.5$ should be printed, whereas "-0.5" means all constraints having $xx[j] \leq bux[j] - 0.5$ should be listed.

An empty filter means no filter is applied.

Possible Values:

Any valid file name.

Default value:

""

• `stat_file_name`**Corresponding constant:**

MSK_SPAR_STAT_FILE_NAME

Description:

Statistics file name.

Possible Values:

Any valid file name.

Default value:

""

• `stat_key`**Corresponding constant:**

MSK_SPAR_STAT_KEY

Description:

Key used when writing the summary file.

Possible Values:

Any valid XML string.

Default value:

""

• `stat_name`**Corresponding constant:**

MSK_SPAR_STAT_NAME

Description:

Name used when writing the statistics file.

Possible Values:

Any valid XML string.

Default value:

""

• `write_lp_gen_var_name`**Corresponding constant:**

MSK_SPAR_WRITE_LP_GEN_VAR_NAME

Description:

Sometimes when an LP file is written additional variables must be inserted. They will have the prefix denoted by this parameter.

Possible Values:

Any valid string.

Default value:

`"xmskgen"`

Appendix F

Symbolic constants

F.1 Constraint or variable access modes

Value	Name	Description
0	MSK_ACC_VAR	Access data by columns (variable oriented)
1	MSK_ACC_CON	Access data by rows (constraint oriented)

F.2 Function opcode

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

continued on next page

continued from previous page	
3	MSK_ADOP_DIV Divide two operands.
6	MSK_ADOP_LOG Logarithm function of one operand.

F.3 Function operand type

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

F.4 Basis identification

Value	Name Description
1	MSK_BI_ALWAYS Basis identification is always performed even if the interior-point optimizer terminates abnormally.
2	MSK_BI_NO_ERROR Basis identification is performed if the interior-point optimizer terminates without an error.
0	MSK_BI_NEVER Never do basis identification.
3	MSK_BI_IF_FEASIBLE Basis identification is not performed if the interior-point optimizer terminates with a problem status saying that the problem is primal or dual infeasible.
4	MSK_BI_OTHER Try another BI method.

F.5 Bound keys

Value	Name Description
2	MSK_BK_FX The constraint or variable is fixed.
0	MSK_BK_LO The constraint or variable has a finite lower bound and an infinite upper bound.
3	MSK_BK_FR The constraint or variable is free.
1	MSK_BK_UP The constraint or variable has an infinite lower bound and an finite upper bound.
4	MSK_BK_RA The constraint or variable is ranged.

F.6 Specifies the branching direction.

Value	Name Description
2	MSK_BRANCH_DIR_DOWN The mixed-integer optimizer always chooses the down branch first.
1	MSK_BRANCH_DIR_UP The mixed-integer optimizer always chooses the up branch first.
0	MSK_BRANCH_DIR_FREE The mixed-integer optimizer decides which branch to choose.

F.7 Progress call-back codes

Value	Name Description
44	MSK_CALLBACK_END_INTPNT The call-back function is called when the interior-point optimizer is terminated.
21	MSK_CALLBACK_BEGIN_PRIMAL_DUAL_SIMPLEX_BI

continued on next page

continued from previous page

	The call-back function is called from within the basis identification procedure when the primal-dual simplex clean-up phase is started.
48	MSK_CALLBACK_END_NETWORK_PRIMAL_SIMPLEX The call-back function is called when the primal network simplex optimizer is terminated.
99	MSK_CALLBACK_READ_ADD_CONS A chunk of constraints has been read from a problem file.
115	MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX_BI The call-back function is called from within the basis identification procedure at an intermediate point in the primal simplex clean-up phase. The frequency of the call-backs is controlled by the MSK_IPAR_LOG_SIM_FREQ parameter.
46	MSK_CALLBACK_END_MIO The call-back function is called when the mixed-integer optimizer is terminated.
13	MSK_CALLBACK_BEGIN_NETWORK_DUAL_SIMPLEX The call-back function is called when the dual network simplex optimizer is started.
35	MSK_CALLBACK_END_CONCURRENT Concurrent optimizer is terminated.
93	MSK_CALLBACK_NEW_INT_MIO The call-back function is called after a new integer solution has been located by the mixed-integer optimizer.
88	MSK_CALLBACK_IM_PRIMAL_SIMPLEX The call-back function is called at an intermediate point in the primal simplex optimizer.
64	MSK_CALLBACK_END_SIMPLEX_NETWORK_DETECT The call-back function is called when the network detection procedure is terminated.
47	MSK_CALLBACK_END_NETWORK_DUAL_SIMPLEX The call-back function is called when the dual network simplex optimizer is terminated.
72	MSK_CALLBACK_IM_INTPNT The call-back function is called at an intermediate stage within the interior-point optimizer where the information database has not been updated.
68	MSK_CALLBACK_IM_DUAL_BI The call-back function is called from within the basis identification procedure at an intermediate point in the dual phase.

continued on next page

continued from previous page	
8	MSK_CALLBACK_BEGIN_FULL_CONVEXITY_CHECK Begin full convexity check.
3	MSK_CALLBACK_BEGIN_DUAL_BI The call-back function is called from within the basis identification procedure when the dual phase is started.
4	MSK_CALLBACK_BEGIN_DUAL_SENSITIVITY Dual sensitivity analysis is started.
79	MSK_CALLBACK_IM_MIO_PRIMAL_SIMPLEX The call-back function is called at an intermediate point in the mixed-integer optimizer while running the primal simplex optimizer.
19	MSK_CALLBACK_BEGIN_PRIMAL_BI The call-back function is called from within the basis identification procedure when the primal phase is started.
100	MSK_CALLBACK_READ_ADD_QNZ A chunk of Q non-zeros has been read from a problem file.
1	MSK_CALLBACK_BEGIN_CONCURRENT Concurrent optimizer is started.
104	MSK_CALLBACK_UPDATE_DUAL_BI The call-back function is called from within the basis identification procedure at an intermediate point in the dual phase.
70	MSK_CALLBACK_IM_DUAL_SIMPLEX The call-back function is called at an intermediate point in the dual simplex optimizer.
11	MSK_CALLBACK_BEGIN_LICENSE_WAIT Begin waiting for license.
81	MSK_CALLBACK_IM_NETWORK_PRIMAL_SIMPLEX The call-back function is called at an intermediate point in the primal network simplex optimizer.
49	MSK_CALLBACK_END_NETWORK_SIMPLEX The call-back function is called when the simplex network optimizer is terminated.
32	MSK_CALLBACK_CONIC The call-back function is called from within the conic optimizer after the information database has been updated.
89	MSK_CALLBACK_IM_QO_REFORMULATE The call-back function is called at an intermediate stage of the QP to SOCP reformulation.
2	MSK_CALLBACK_BEGIN_CONIC The call-back function is called when the conic optimizer is started.

continued on next page

continued from previous page

106	<code>MSK_CALLBACK_UPDATE_DUAL_SIMPLEX_BI</code> 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 <code>MSK_IPAR_LOG_SIM_FREQ</code> parameter.
51	<code>MSK_CALLBACK_END_OPTIMIZER</code> The call-back function is called when the optimizer is terminated.
110	<code>MSK_CALLBACK_UPDATE_PRESOLVE</code> The call-back function is called from within the presolve procedure.
90	<code>MSK_CALLBACK_IM_SIMPLEX</code> The call-back function is called from within the simplex optimizer at an intermediate point.
102	<code>MSK_CALLBACK_READ_OPF</code> The call-back function is called from the OPF reader.
73	<code>MSK_CALLBACK_IM_LICENSE_WAIT</code> MOSEK is waiting for a license.
15	<code>MSK_CALLBACK_BEGIN_NETWORK_SIMPLEX</code> The call-back function is called when the simplex network optimizer is started.
36	<code>MSK_CALLBACK_END_CONIC</code> The call-back function is called when the conic optimizer is terminated.
107	<code>MSK_CALLBACK_UPDATE_NETWORK_DUAL_SIMPLEX</code> The call-back function is called in the dual network simplex optimizer.
26	<code>MSK_CALLBACK_BEGIN_QCQO_REFORMULATE</code> Begin QCQO reformulation.
38	<code>MSK_CALLBACK_END_DUAL_SENSITIVITY</code> Dual sensitivity analysis is terminated.
59	<code>MSK_CALLBACK_END_PRIMAL_SIMPLEX_BI</code> The call-back function is called from within the basis identification procedure when the primal clean-up phase is terminated.
101	<code>MSK_CALLBACK_READ_ADD_VARS</code> A chunk of variables has been read from a problem file.
103	<code>MSK_CALLBACK_READ_OPF_SECTION</code> A chunk of Q non-zeros has been read from a problem file.
74	<code>MSK_CALLBACK_IM_LU</code> The call-back function is called from within the LU factorization procedure at an intermediate point.
41	<code>MSK_CALLBACK_END_DUAL_SIMPLEX_BI</code>

continued on next page

continued from previous page

	The call-back function is called from within the basis identification procedure when the dual clean-up phase is terminated.
45	MSK_CALLBACK_END_LICENSE_WAIT End waiting for license.
84	MSK_CALLBACK_IM_PRESOLVE The call-back function is called from within the presolve procedure at an intermediate stage.
5	MSK_CALLBACK_BEGIN_DUAL_SETUP_BI The call-back function is called when the dual BI phase is started.
43	MSK_CALLBACK_END_INFEAS_ANA The call-back function is called when the infeasibility analyzer is terminated.
92	MSK_CALLBACK_INTPNT The call-back function is called from within the interior-point optimizer after the information database has been updated.
111	MSK_CALLBACK_UPDATE_PRIMAL_BI The call-back function is called from within the basis identification procedure at an intermediate point in the primal phase.
94	MSK_CALLBACK_NONCOVEX The call-back function is called from within the nonconvex optimizer after the information database has been updated.
113	MSK_CALLBACK_UPDATE_PRIMAL_DUAL_SIMPLEX_BI The call-back function is called from within the basis identification procedure at an intermediate point in the primal-dual simplex clean-up phase. The frequency of the call-backs is controlled by the MSK_IPAR_LOG_SIM_FREQ parameter.
109	MSK_CALLBACK_UPDATE_NONCONVEX The call-back function is called at an intermediate stage within the nonconvex optimizer where the information database has been updated.
37	MSK_CALLBACK_END_DUAL_BI The call-back function is called from within the basis identification procedure when the dual phase is terminated.
61	MSK_CALLBACK_END_READ MOSEK has finished reading a problem file.
98	MSK_CALLBACK_READ_ADD_CONES A chunk of cones has been read from a problem file.
25	MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX_BI

continued on next page

	continued from previous page
	The call-back function is called from within the basis identification procedure when the primal simplex clean-up phase is started.
30	<code>MSK_CALLBACK_BEGIN_SIMPLEX_NETWORK_DETECT</code> The call-back function is called when the network detection procedure is started.
97	<code>MSK_CALLBACK_READ_ADD_ANZ</code> A chunk of A non-zeros has been read from a problem file.
86	<code>MSK_CALLBACK_IM_PRIMAL_DUAL_SIMPLEX</code> The call-back function is called at an intermediate point in the primal-dual simplex optimizer.
114	<code>MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX</code> The call-back function is called in the primal simplex optimizer.
33	<code>MSK_CALLBACK_DUAL_SIMPLEX</code> The call-back function is called from within the dual simplex optimizer.
71	<code>MSK_CALLBACK_IM_FULL_CONVEXITY_CHECK</code> The call-back function is called at an intermediate stage of the full convexity check.
95	<code>MSK_CALLBACK_PRIMAL_SIMPLEX</code> The call-back function is called from within the primal simplex optimizer.
16	<code>MSK_CALLBACK_BEGIN_NONCONVEX</code> The call-back function is called when the nonconvex optimizer is started.
91	<code>MSK_CALLBACK_IM_SIMPLEX_BI</code> 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 <code>MSK_IPAR_LOG_SIM_FREQ</code> parameter.
6	<code>MSK_CALLBACK_BEGIN_DUAL_SIMPLEX</code> The call-back function is called when the dual simplex optimizer started.
24	<code>MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX</code> The call-back function is called when the primal simplex optimizer is started.
50	<code>MSK_CALLBACK_END_NONCONVEX</code> The call-back function is called when the nonconvex optimizer is terminated.
23	<code>MSK_CALLBACK_BEGIN_PRIMAL_SETUP_BI</code>

continued on next page

	continued from previous page
	The call-back function is called when the primal BI setup is started.
17	MSK_CALLBACK_BEGIN_OPTIMIZER
	The call-back function is called when the optimizer is started.
27	MSK_CALLBACK_BEGIN_READ
	MOSEK has started reading a problem file.
82	MSK_CALLBACK_IM_NONCONVEX
	The call-back function is called at an intermediate stage within the nonconvex optimizer where the information database has not been updated.
58	MSK_CALLBACK_END_PRIMAL_SIMPLEX
	The call-back function is called when the primal simplex optimizer is terminated.
55	MSK_CALLBACK_END_PRIMAL_DUAL_SIMPLEX_BI
	The call-back function is called from within the basis identification procedure when the primal-dual clean-up phase is terminated.
66	MSK_CALLBACK_IM_BI
	The call-back function is called from within the basis identification procedure at an intermediate point.
80	MSK_CALLBACK_IM_NETWORK_DUAL_SIMPLEX
	The call-back function is called at an intermediate point in the dual network simplex optimizer.
39	MSK_CALLBACK_END_DUAL_SETUP_BI
	The call-back function is called when the dual BI phase is terminated.
34	MSK_CALLBACK_END_BI
	The call-back function is called when the basis identification procedure is terminated.
57	MSK_CALLBACK_END_PRIMAL_SETUP_BI
	The call-back function is called when the primal BI setup is terminated.
31	MSK_CALLBACK_BEGIN_WRITE
	MOSEK has started writing a problem file.
63	MSK_CALLBACK_END_SIMPLEX_BI
	The call-back function is called from within the basis identification procedure when the simplex clean-up phase is terminated.
56	MSK_CALLBACK_END_PRIMAL_SENSITIVITY
	Primal sensitivity analysis is terminated.
28	MSK_CALLBACK_BEGIN_SIMPLEX
	The call-back function is called when the simplex optimizer is started.
52	MSK_CALLBACK_END_PRESOLVE

continued on next page

	continued from previous page
	The call-back function is called when the presolve is completed.
96	<code>MSK_CALLBACK_QCONE</code> The call-back function is called from within the Qcone optimizer.
9	<code>MSK_CALLBACK_BEGIN_INFEAS_ANA</code> The call-back function is called when the infeasibility analyzer is started.
20	<code>MSK_CALLBACK_BEGIN_PRIMAL_DUAL_SIMPLEX</code> The call-back function is called when the primal-dual simplex optimizer is started.
22	<code>MSK_CALLBACK_BEGIN_PRIMAL_SENSITIVITY</code> Primal sensitivity analysis is started.
7	<code>MSK_CALLBACK_BEGIN_DUAL_SIMPLEX_BI</code> The call-back function is called from within the basis identification procedure when the dual simplex clean-up phase is started.
60	<code>MSK_CALLBACK_END_QCQO_REFORMULATE</code> End QCQO reformulation.
87	<code>MSK_CALLBACK_IM_PRIMAL_SENSITIVITY</code> The call-back function is called at an intermediate stage of the primal sensitivity analysis.
65	<code>MSK_CALLBACK_END_WRITE</code> MOSEK has finished writing a problem file.
40	<code>MSK_CALLBACK_END_DUAL_SIMPLEX</code> The call-back function is called when the dual simplex optimizer is terminated.
112	<code>MSK_CALLBACK_UPDATE_PRIMAL_DUAL_SIMPLEX</code> The call-back function is called in the primal-dual simplex optimizer.
29	<code>MSK_CALLBACK_BEGIN_SIMPLEX_BI</code> The call-back function is called from within the basis identification procedure when the simplex clean-up phase is started.
10	<code>MSK_CALLBACK_BEGIN_INTPNT</code> The call-back function is called when the interior-point optimizer is started.
69	<code>MSK_CALLBACK_IM_DUAL_SENSITIVITY</code> The call-back function is called at an intermediate stage of the dual sensitivity analysis.
62	<code>MSK_CALLBACK_END_SIMPLEX</code> The call-back function is called when the simplex optimizer is terminated.
53	<code>MSK_CALLBACK_END_PRIMAL_BI</code>

continued on next page

continued from previous page

	The call-back function is called from within the basis identification procedure when the primal phase is terminated.
75	<code>MSK_CALLBACK_IM_MIO</code> The call-back function is called at an intermediate point in the mixed-integer optimizer.
105	<code>MSK_CALLBACK_UPDATE_DUAL_SIMPLEX</code> The call-back function is called in the dual simplex optimizer.
77	<code>MSK_CALLBACK_IM_MIO_INTPNT</code> The call-back function is called at an intermediate point in the mixed-integer optimizer while running the interior-point optimizer.
54	<code>MSK_CALLBACK_END_PRIMAL_DUAL_SIMPLEX</code> The call-back function is called when the primal-dual simplex optimizer is terminated.
67	<code>MSK_CALLBACK_IM_CONIC</code> The call-back function is called at an intermediate stage within the conic optimizer where the information database has not been updated.
78	<code>MSK_CALLBACK_IM_MIO_PRESOLVE</code> The call-back function is called at an intermediate point in the mixed-integer optimizer while running the presolve.
0	<code>MSK_CALLBACK_BEGIN_BI</code> The basis identification procedure has been started.
76	<code>MSK_CALLBACK_IM_MIO_DUAL_SIMPLEX</code> The call-back function is called at an intermediate point in the mixed-integer optimizer while running the dual simplex optimizer.
116	<code>MSK_CALLBACK_WRITE_OPF</code> The call-back function is called from the OPF writer.
108	<code>MSK_CALLBACK_UPDATE_NETWORK_PRIMAL_SIMPLEX</code> The call-back function is called in the primal network simplex optimizer.
42	<code>MSK_CALLBACK_END_FULL_CONVEXITY_CHECK</code> End full convexity check.
83	<code>MSK_CALLBACK_IM_ORDER</code> The call-back function is called from within the matrix ordering procedure at an intermediate point.
85	<code>MSK_CALLBACK_IM_PRIMAL_BI</code> The call-back function is called from within the basis identification procedure at an intermediate point in the primal phase.
18	<code>MSK_CALLBACK_BEGIN_PRESOLVE</code> The call-back function is called when the presolve is started.

continued on next page

continued from previous page		
12	<code>MSK_CALLBACK_BEGIN_MIO</code>	The call-back function is called when the mixed-integer optimizer is started.
14	<code>MSK_CALLBACK_BEGIN_NETWORK_PRIMAL_SIMPLEX</code>	The call-back function is called when the primal network simplex optimizer is started.

F.8 Types of convexity checks.

Value	Name	Description
1	<code>MSK_CHECK_CONVEXITY_SIMPLE</code>	Perform simple and fast convexity check.
0	<code>MSK_CHECK_CONVEXITY_NONE</code>	No convexity check.
2	<code>MSK_CHECK_CONVEXITY_FULL</code>	Perform a full convexity check.

F.9 Compression types

Value	Name	Description
2	<code>MSK_COMPRESS_GZIP</code>	The type of compression used is gzip compatible.
0	<code>MSK_COMPRESS_NONE</code>	No compression is used.
1	<code>MSK_COMPRESS_FREE</code>	The type of compression used is chosen automatically.

F.10 Cone types

Value	Name	Description
0	<code>MSK_CT_QUAD</code>	The cone is a quadratic cone.

continued on next page

continued from previous page	
1	MSK_CT_RQUAD The cone is a rotated quadratic cone.

F.11 CPU type

Value	Name Description
8	MSK_CPU_POWERPC_G5 A G5 PowerPC CPU.
9	MSK_CPU_INTEL_PM An Intel PM cpu.
1	MSK_CPU_GENERIC An generic CPU type for the platform
0	MSK_CPU_UNKNOWN An unknown CPU.
7	MSK_CPU_AMD_OPTERON An AMD Opteron (64 bit).
6	MSK_CPU_INTEL_ITANIUM2 An Intel Itanium2.
4	MSK_CPU_AMD_ATHLON An AMD Athlon.
5	MSK_CPU_HP_PARISC20 An HP PA RISC version 2.0 CPU.
3	MSK_CPU_INTEL_P4 An Intel Pentium P4 or Intel Xeon.
2	MSK_CPU_INTEL_P3 An Intel Pentium P3.
10	MSK_CPU_INTEL_CORE2 An Intel CORE2 cpu.

F.12 Data format types

Value	Name Description
5	MSK_DATA_FORMAT_XML The data file is an XML formatted file.

continued on next page

continued from previous page	
6	MSK_DATA_FORMAT_FREE_MPS The data data a free MPS formatted file.
0	MSK_DATA_FORMAT_EXTENSION The file extension is used to determine the data file format.
1	MSK_DATA_FORMAT_MPS The data file is MPS formatted.
2	MSK_DATA_FORMAT_LP The data file is LP formatted.
3	MSK_DATA_FORMAT_MBT The data file is a MOSEK binary task file.
4	MSK_DATA_FORMAT_OP The data file is an optimization problem formatted file.

F.13 Double information items

Value	Name Description
13	MSK_DINF_INTPNT_PRIMAL_FEAS Primal feasibility measure reported by the interior-point or Qcone optimizers. (For the interior-point optimizer this measure does not directly related to the original problem because a homogeneous model is employed).
58	MSK_DINF_SOL_ITR_MAX_PCNI Maximal primal cone infeasibility in the interior-point solution. Updated at the end of the optimization.
32	MSK_DINF_RD_TIME Time spent reading the data file.
28	MSK_DINF_PRESOLVE_ELI_TIME Total time spent in the eliminator since the presolve was invoked.
22	MSK_DINF_MIO_OPTIMIZER_TIME Time spent in the optimizer while solving the relaxtions.
10	MSK_DINF_INTPNT_FACTOR_NUM_FLOPS An estimate of the number of flops used in the factorization.
25	MSK_DINF_MIO_TIME Time spent in the mixed-integer optimizer.
4	MSK_DINF_BI_DUAL_TIME Time spent within the dual phase basis identification procedure since its invocation.

continued on next page

continued from previous page

37	MSK_DINF_SIM_NETWORK_TIME Time spent in the network simplex optimizer since invoking it.
30	MSK_DINF_PRESOLVE_TIME Total time (in seconds) spent in the presolve since it was invoked.
29	MSK_DINF_PRESOLVE_LINDEP_TIME Total time spent in the linear dependency checker since the presolve was invoked.
33	MSK_DINF_SIM_DUAL_TIME Time spent in the dual simplex optimizer since invoking it.
60	MSK_DINF_SOL_ITR_MAX_PINTI Maximal primal integer infeasibility in the interior-point solution. Updated at the end of the optimization.
38	MSK_DINF_SIM_OBJ Objective value reported by the simplex optimizer.
21	MSK_DINF_MIO_OBJ_REL_GAP 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 parameter **MSK_DPAR.MIO_REL_GAP_CONST**. Otherwise it has the value -1.0.

48	MSK_DINF_SOL_BAS_PRIMAL_OBJ Primal objective value of the basic solution. Updated at the end of the optimization.
17	MSK_DINF_MIO_HEURISTIC_TIME Time spent in the optimizer while solving the relaxations.
57	MSK_DINF_SOL_ITR_MAX_PBI Maximal primal bound infeasibility in the interior-point solution. Updated at the end of the optimization.
45	MSK_DINF_SOL_BAS_MAX_PBI Maximal primal bound infeasibility in the basic solution. Updated at the end of the optimization.
27	MSK_DINF_OPTIMIZER_TIME Total time spent in the optimizer since it was invoked.
55	MSK_DINF_SOL_ITR_MAX_DCNI

continued on next page

continued from previous page	
	Maximal dual cone infeasibility in the interior-point solution. Updated at the end of the optimization.
24	MSK_DINF_MIO_ROOT_PRESOLVE_TIME Time spent in while presolveing the root relaxation.
9	MSK_DINF_INTPNT_DUAL_OBJ Dual objective value reported by the interior-point or Qcone optimizer.
15	MSK_DINF_INTPNT_TIME Time spent within the interior-point optimizer since its invocation.
16	MSK_DINF_MIO_CONSTRUCT_SOLUTION_OBJ If MOSEK has successfully constructed an integer feasible solution, then this item contains the optimal objective value corresponding to the feasible solution.
34	MSK_DINF_SIM_FEAS Feasibility measure reported by the simplex optimizer.
56	MSK_DINF_SOL_ITR_MAX_DEQI Maximal dual equality infeasibility in the interior-point solution. Updated at the end of the optimization.
40	MSK_DINF_SIM_PRIMAL_TIME Time spent in the primal simplex optimizer since invoking it.
41	MSK_DINF_SIM_TIME Time spent in the simplex optimizer since invoking it.
36	MSK_DINF_SIM_NETWORK_PRIMAL_TIME Time spent in the primal network simplex optimizer since invoking it.
47	MSK_DINF_SOL_BAS_MAX_PINTI Maximal primal integer infeasibility in the basic solution. Updated at the end of the optimization.
51	MSK_DINF_SOL_INT_MAX_PINTI Maximal primal integer infeasibility in the integer solution. Updated at the end of the optimization.
3	MSK_DINF_BI_CLEAN_TIME Time spent within the clean-up phase of the basis identification procedure since its invocation.
31	MSK_DINF_QCQO_REFORMULATE_TIME Time spent with QP reformulation.
53	MSK_DINF_SOL_ITR_DUAL_OBJ Dual objective value of the interior-point solution. Updated at the end of the optimization.

continued on next page

continued from previous page	
49	MSK_DINF_SOL_INT_MAX_PBI Maximal primal bound infeasibility in the integer solution. Updated at the end of the optimization.
8	MSK_DINF_INTPNT_DUAL_FEAS Dual feasibility measure reported by the interior-point and Qcone optimizer. (For the interior-point optimizer this measure does not directly related to the original problem because a homogeneous model is employed.)
7	MSK_DINF_CONCURRENT_TIME Time spent within the concurrent optimizer since its invocation.
11	MSK_DINF_INTPNT_KAP_DIV_TAU This measure should converge to zero if the problem has a primal-dual optimal solution or to infinity if problem is (strictly) primal or dual infeasible. In case the measure is converging towards a positive but bounded constant the problem is usually ill-posed.
50	MSK_DINF_SOL_INT_MAX_PEQI Maximal primal equality infeasibility in the basic solution. Updated at the end of the optimization.
61	MSK_DINF_SOL_ITR_PRIMAL_OBJ Primal objective value of the interior-point solution. Updated at the end of the optimization.
35	MSK_DINF_SIM_NETWORK_DUAL_TIME Time spent in the dual network simplex optimizer since invoking it.
20	MSK_DINF_MIO_OBJ_INT The primal objective value corresponding to the best integer feasible solution. Please note that at least one integer feasible solution must have located i.e. check MSK_IINF_MIO_NUM_INT_SOLUTIONS .
26	MSK_DINF_MIO_USER_OBJ_CUT If the objective cut is used, then this information item has the value of the cut.
43	MSK_DINF_SOL_BAS_MAX_DBI Maximal dual bound infeasibility in the basic solution. Updated at the end of the optimization.
46	MSK_DINF_SOL_BAS_MAX_PEQI Maximal primal equality infeasibility in the basic solution. Updated at the end of the optimization.
19	MSK_DINF_MIO_OBJ_BOUND

continued on next page

continued from previous page	
	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 <code>MSK_IINF_MIO_NUM_RELAX</code> is strictly positive.
0	<code>MSK_DINF_BI_CLEAN_DUAL_TIME</code> Time spent within the dual clean-up optimizer of the basis identification procedure since its invocation.
6	<code>MSK_DINF_BI_TIME</code> Time spent within the basis identification procedure since its invocation.
42	<code>MSK_DINF_SOL_BAS_DUAL_OBJ</code> Dual objective value of the basic solution. Updated at the end of the optimization.
1	<code>MSK_DINF_BI_CLEAN_PRIMAL_DUAL_TIME</code> Time spent within the primal-dual clean-up optimizer of the basis identification procedure since its invocation.
14	<code>MSK_DINF_INTPNT_PRIMAL_OBJ</code> Primal objective value reported by the interior-point or Qcone optimizer.
12	<code>MSK_DINF_INTPNT_ORDER_TIME</code> Order time (in seconds).
52	<code>MSK_DINF_SOL_INT_PRIMAL_OBJ</code> Primal objective value of the integer solution. Updated at the end of the optimization.
5	<code>MSK_DINF_BI_PRIMAL_TIME</code> Time spent within the primal phase of the basis identification procedure since its invocation.
18	<code>MSK_DINF_MIO_OBJ_ABS_GAP</code> 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.
44	<code>MSK_DINF_SOL_BAS_MAX_DEQI</code> Maximal dual equality infeasibility in the basic solution. Updated at the end of the optimization.
59	<code>MSK_DINF_SOL_ITR_MAX_PEQI</code>
continued on next page	

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

F.14 Double parameters

Value	Name Description
40	MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH If the lower objective cut is less than the value of this parameter value, then the lower objective cut i.e. MSK_DPAR_LOWER_OBJ_CUT is treated as $-\infty$.
43	MSK_DPAR_MIO_MAX_TIME This parameter limits the maximum time spent by the mixed-integer optimizer. A negative number means infinity.
2	MSK_DPAR_BASIS_TOL_S Maximum absolute dual bound violation in an optimal basic solution.
60	MSK_DPAR_PRESOLVE_TOL_S Absolute zero tolerance employed for s_i in the presolve.
65	MSK_DPAR_UPPER_OBJ_CUT If either a primal or dual feasible solution is found proving that the optimal objective value is outside, [MSK_DPAR_LOWER_OBJ_CUT , MSK_DPAR_UPPER_OBJ_CUT], then MOSEK is terminated.
16	MSK_DPAR_INTPNT_CO_TOL_DFEAS Dual feasibility tolerance used by the conic interior-point optimizer.
8	MSK_DPAR_DATA_TOL_AIJ_LARGE An element in A which is larger than this value in absolute size causes a warning message to be printed.

continued on next page

continued from previous page

49	<code>MSK_DPAR_MIO_TOL_ABS_GAP</code> Absolute optimality tolerance employed by the mixed-integer optimizer.
66	<code>MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH</code> If the upper objective cut is greater than the value of this value parameter, then the the upper objective cut <code>MSK_DPAR_UPPER_OBJ_CUT</code> is treated as ∞ .
50	<code>MSK_DPAR_MIO_TOL_ABS_RELAX_INT</code> 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.
56	<code>MSK_DPAR_NONCONVEX_TOL_OPT</code> Optimality tolerance used by the nonconvex optimizer.
55	<code>MSK_DPAR_NONCONVEX_TOL_FEAS</code> Feasibility tolerance used by the nonconvex optimizer.
64	<code>MSK_DPAR_SIMPLEX_ABS_TOL_PIV</code> Absolute pivot tolerance employed by the simplex optimizers.
42	<code>MSK_DPAR_MIO_HEURISTIC_TIME</code> Minimum amount of time to be used in the heuristic search for a good feasible integer solution. A negative values implies that the optimizer decides the amount of time to be spent in the heuristic.
5	<code>MSK_DPAR_CHECK_CONVEXITY_REL_TOL</code> 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:
	$d_i \leq - Q_{ii} * \text{check_convexity_rel_tol}$
61	<code>MSK_DPAR_PRESOLVE_TOL_X</code> Absolute zero tolerance employed for x_j in the presolve.
24	<code>MSK_DPAR_INTPNT_NL_TOL_MU_RED</code> Relative complementarity gap tolerance.

continued on next page

continued from previous page

46	MSK_DPAR_MIO_NEAR_TOL_REL_GAP
	The mixed-integer optimizer is terminated when this tolerance is satisfied. This termination criteria is delayed. See MSK_DPAR_MIO_DISABLE_TERM_TIME for details.
6	MSK_DPAR_DATA_TOL_AIJ
	Absolute zero tolerance for elements in A . If any value A_{ij} is smaller than this parameter in absolute terms MOSEK will treat the values as zero and generate a warning.
15	MSK_DPAR_FEASREPAIR_TOL
	Tolerance for constraint enforcing upper bound on sum of weighted violations in feasibility repair.
30	MSK_DPAR_INTPNT_TOL_DSAFE
	Controls the initial dual starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly.
51	MSK_DPAR_MIO_TOL_FEAS
	Feasibility tolerance for mixed integer solver. Any solution with maximum infeasibility below this value will be considered feasible.
31	MSK_DPAR_INTPNT_TOL_INFEAS
	Controls when the optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.
25	MSK_DPAR_INTPNT_NL_TOL_NEAR_REL
	If the MOSEK nonlinear interior-point optimizer cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.
57	MSK_DPAR_OPTIMIZER_MAX_TIME
	Maximum amount of time the optimizer is allowed to spent on the optimization. A negative number means infinity.
14	MSK_DPAR_DATA_TOL_X
	Zero tolerance for constraints and variables i.e. if the distance between the lower and upper bound is less than this value, then the lower and lower bound is considered identical.
0	MSK_DPAR_ANA_SOL_INFEAS_TOL
	If a constraint violates its bound with an amount larger than this value, the constraint name, index and violation will be printed by the solution analyzer.
47	MSK_DPAR_MIO_REL_ADD_CUT_LIMITED

continued on next page

continued from previous page

	Controls how many cuts the mixed-integer optimizer is allowed to add to the problem. Let α be the value of this parameter and m the number constraints, then mixed-integer optimizer is allowed to αm cuts.
32	<code>MSK_DPAR_INTPNT_TOL_MU_RED</code> Relative complementarity gap tolerance.
18	<code>MSK_DPAR_INTPNT_CO_TOL_MU_RED</code> Relative complementarity gap tolerance feasibility tolerance used by the conic interior-point optimizer.
21	<code>MSK_DPAR_INTPNT_CO_TOL_REL_GAP</code> Relative gap termination tolerance used by the conic interior-point optimizer.
39	<code>MSK_DPAR_LOWER_OBJ_CUT</code> If either a primal or dual feasible solution is found proving that the optimal objective value is outside, the interval <code>[MSK_DPAR_LOWER_OBJ_CUT, MSK_DPAR_UPPER_OBJ_CUT]</code> , then MOSEK is terminated.
41	<code>MSK_DPAR_MIO_DISABLE_TERM_TIME</code> The termination criteria governed by <ul style="list-style-type: none"> • <code>MSK_IPAR_MIO_MAX_NUM_RELAXS</code> • <code>MSK_IPAR_MIO_MAX_NUM_BRANCHES</code> • <code>MSK_DPAR_MIO_NEAR_TOL_ABS_GAP</code> • <code>MSK_DPAR_MIO_NEAR_TOL_REL_GAP</code> is disabled the first n seconds. This parameter specifies the number n . A negative value is identical to infinity i.e. the termination criteria are never checked.
37	<code>MSK_DPAR_INTPNT_TOL_REL_STEP</code> Relative step size to the boundary for linear and quadratic optimization problems.
54	<code>MSK_DPAR_MIO_TOL_X</code> Absolute solution tolerance used in mixed-integer optimizer.
11	<code>MSK_DPAR_DATA_TOL_C_HUGE</code> An element in c which is larger than the value of this parameter in absolute terms is considered to be huge and generates an error.
59	<code>MSK_DPAR_PRESOLVE_TOL_LIN_DEP</code>

continued on next page

continued from previous page

	Controls when a constraint is determined to be linearly dependent.
63	MSK_DPAR_SIM_LU_TOL_REL_PIV Relative pivot tolerance employed when computing the LU factorization of the basis in the simplex optimizers and in the basis identification procedure. A value closer to 1.0 generally improves numerical stability but typically also implies an increase in the computational work.
12	MSK_DPAR_DATA_TOL_CJ_LARGE An element in c which is larger than this value in absolute terms causes a warning message to be printed.
28	MSK_DPAR_INTPNT_NL_TOL_REL_STEP Relative step size to the boundary for general nonlinear optimization problems.
38	MSK_DPAR_INTPNT_TOL_STEP_SIZE If the step size falls below the value of this parameter, then the interior-point optimizer assumes that it is stalled. If it does not make any progress.
34	MSK_DPAR_INTPNT_TOL_PFEAS Primal feasibility tolerance used for linear and quadratic optimization problems.
1	MSK_DPAR_BASIS_REL_TOL_S Maximum relative dual bound violation allowed in an optimal basic solution.
17	MSK_DPAR_INTPNT_CO_TOL_INFEAS Controls when the conic interior-point optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.
48	MSK_DPAR_MIO_REL_GAP_CONST This value is used to compute the relative gap for the solution to an integer optimization problem.
58	MSK_DPAR_PREOLVE_TOL_AIJ Absolute zero tolerance employed for a_{ij} in the presolve.
44	MSK_DPAR_MIO_MAX_TIME_APRX_OPT Number of seconds spent by the mixed-integer optimizer before the MSK_DPAR_MIO_TOL_REL_RELAX_INT is applied.
33	MSK_DPAR_INTPNT_TOL_PATH

continued on next page

continued from previous page	
	Controls how close the interior-point optimizer follows the central path. A large value of this parameter means the central is followed very closely. On numerical unstable problems it may be worthwhile to increase this parameter.
22	MSK_DPAR_INTPNT_NL_MERIT_BAL Controls if the complementarity and infeasibility is converging to zero at about equal rates.
3	MSK_DPAR_BASIS_TOL_X Maximum absolute primal bound violation allowed in an optimal basic solution.
36	MSK_DPAR_INTPNT_TOL_REL_GAP Relative gap termination tolerance.
7	MSK_DPAR_DATA_TOL_AIJ_HUGE An element in A which is larger than this value in absolute size causes an error.
10	MSK_DPAR_DATA_TOL_BOUND_WRN If a bound value is larger than this value in absolute size, then a warning message is issued.
9	MSK_DPAR_DATA_TOL_BOUND_INF Any bound which in absolute value is greater than this parameter is considered infinite.
35	MSK_DPAR_INTPNT_TOL_PSAFE Controls the initial primal starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly and/or the constraint or variable bounds are very large, then it may be worthwhile to increase this value.
19	MSK_DPAR_INTPNT_CO_TOL_NEAR_REL If MOSEK cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.
4	MSK_DPAR_CALLBACK_FREQ Controls the time between calls to the progress call-back function. Hence, if the value of this parameter is for example 10, then the call-back is called approximately each 10 seconds. A negative value is equivalent to infinity. In general frequent call-backs may hurt the performance.
26	MSK_DPAR_INTPNT_NL_TOL_PFEAS Primal feasibility tolerance used when a nonlinear model is solved.

continued on next page

continued from previous page	
23	MSK_DPAR_INTPNT_NL_TOL_DFEAS Dual feasibility tolerance used when a nonlinear model is solved.
52	MSK_DPAR_MIO_TOL_REL_GAP Relative optimality tolerance employed by the mixed-integer optimizer.
29	MSK_DPAR_INTPNT_TOL_DFEAS Dual feasibility tolerance used for linear and quadratic optimization problems.
45	MSK_DPAR_MIO_NEAR_TOL_ABS_GAP Relaxed absolute optimality tolerance employed by the mixed-integer optimizer. This termination criteria is delayed. See MSK_DPAR_MIO_DISABLE_TERM_TIME for details.
53	MSK_DPAR_MIO_TOL_REL_RELAX_INT 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.
62	MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL This parameter determines when columns are dropped in incomplete cholesky factorization doing reformulation of quadratic problems.
13	MSK_DPAR_DATA_TOL_QIJ Absolute zero tolerance for elements in Q matrices.
27	MSK_DPAR_INTPNT_NL_TOL_REL_GAP Relative gap termination tolerance for nonlinear problems.
20	MSK_DPAR_INTPNT_CO_TOL_PFEAS Primal feasibility tolerance used by the conic interior-point optimizer.

F.15 Feasibility repair types

Value	Name Description
0	MSK_FEASREPAIR_OPTIMIZE_NONE Do not optimize the feasibility repair problem.
2	MSK_FEASREPAIR_OPTIMIZE_COMBINED Minimize with original objective subject to minimal weighted violation of bounds.
1	MSK_FEASREPAIR_OPTIMIZE_PENALTY Minimize weighted sum of violations.

F.16 License feature

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

F.17 Integer information items.

Value	Name	Description
57	MSK_IINF_RD_NUMINTVAR	Number of integer-constrained variables read.
90	MSK_IINF_SOL_BAS_SOLSTA	Solution status of the basic solution. Updated after each optimization.
97	MSK_IINF_STO_NUM_A_TRANSPOSES	Number of times the A matrix is transposed. A large number implies that <code>maxnumanz</code> is too small or an inefficient usage of MOSEK. This will occur in particular if the code alternate between accessing rows and columns of A .
48	MSK_IINF_MIO_TOTAL_NUM_OBJ_CUTS	Number of obj cuts.
88	MSK_IINF_SIM_SOLVE_DUAL	Is non-zero if dual problem is solved.
30	MSK_IINF_MIO_NUMCON	Number of constraints in the problem solved by the mixed-integer optimizer.
53	MSK_IINF_OPT_NUMVAR	Number of variables in the problem solved when the optimizer is called
77	MSK_IINF_SIM_NUMVAR	

continued on next page

	continued from previous page
	Number of variables in the problem solved by the simplex optimizer.
46	MSK_IINF_MIO_TOTAL_NUM_LATTICE_CUTS Number of lattice cuts.
71	MSK_IINF_SIM_NETWORK_PRIMAL_DEG_ITER The number of primal network degenerate iterations.
58	MSK_IINF_RD_NUMQ Number of nonempty Q matrices read.
0	MSK_IINF_ANA_PRO_NUM_CON Number of constraints in the problem.
19	MSK_IINF_INTPNT_FACTOR_NUM_OFFCOL Number of columns in the constraint matrix (or Jacobian) that has an offending structure.
11	MSK_IINF_ANA_PRO_NUM_VAR_INT Number of general integer variables.
69	MSK_IINF_SIM_NETWORK_DUAL_INF_ITER The number of iterations taken with dual infeasibility in the network optimizer.
8	MSK_IINF_ANA_PRO_NUM_VAR_CONT Number of continuous variables.
65	MSK_IINF_SIM_DUAL_ITER Number of dual simplex iterations during the last optimization.
61	MSK_IINF_SIM_DUAL_DEG_ITER The number of dual degenerate iterations.
20	MSK_IINF_INTPNT_ITER Number of interior-point iterations since invoking the interior-point optimizer.
45	MSK_IINF_MIO_TOTAL_NUM_KNAPSUR_COVER_CUTS Number of knapsack cover cuts.
33	MSK_IINF_MIO_TOTAL_NUM_BASIS_CUTS Number of basis cuts.
76	MSK_IINF_SIM_NUMCON Number of constraints in the problem solved by the simplex optimizer.
83	MSK_IINF_SIM_PRIMAL_DUAL_ITER Number of primal dual simplex iterations during the last optimization.
5	MSK_IINF_ANA_PRO_NUM_CON_UP Number of constraints with an upper bound and an infinite lower bound.
36	MSK_IINF_MIO_TOTAL_NUM_CLIQUE_CUTS

continued on next page

continued from previous page	
	Number of clique cuts.
80	MSK_IINF_SIM_PRIMAL_DUAL_HOTSTART If 1 then the primal dual simplex algorithm is solving from an advanced basis.
22	MSK_IINF_INTPNT_SOLVE_DUAL Non-zero if the interior-point optimizer is solving the dual problem.
67	MSK_IINF_SIM_NETWORK_DUAL_HOTSTART If 1 then the dual network simplex algorithm is solving from an advanced basis.
54	MSK_IINF_OPTIMIZE_RESPONSE The reponse code returned by optimize.
93	MSK_IINF_SOL_ITR_PROSTA Problem status of the interior-point solution. Updated after each optimization.
60	MSK_IINF_RD_PROTYPE Problem type.
94	MSK_IINF_SOL_ITR_SOLSTA Solution status of the interior-point solution. Updated after each optimization.
2	MSK_IINF_ANA_PRO_NUM_CON_FR Number of unbounded constraints.
81	MSK_IINF_SIM_PRIMAL_DUAL_HOTSTART_LU If 1 then a valid basis factorization of full rank was located and used by the primal dual simplex algorithm.
31	MSK_IINF_MIO_NUMINT Number of integer variables in the problem solved be the mixed-integer optimizer.
35	MSK_IINF_MIO_TOTAL_NUM_CARDGUB_CUTS Number of cardgub cuts.
38	MSK_IINF_MIO_TOTAL_NUM_CONTRA_CUTS Number of contra cuts.
49	MSK_IINF_MIO_TOTAL_NUM_PLAN_LOC_CUTS Number of loc cuts.
64	MSK_IINF_SIM_DUAL_INF_ITER The number of iterations taken with dual infeasibility.
32	MSK_IINF_MIO_NUMVAR Number of variables in the problem solved be the mixed-integer optimizer.
27	MSK_IINF_MIO_NUM_CUTS
continued on next page	

continued from previous page	
	Number of cuts generated by the mixed-integer optimizer.
23	MSK_IINF_MIO_CONSTRUCT_SOLUTION If this item has the value 0, then MOSEK did not try to construct an initial integer feasible solution. If the item has a positive value, then MOSEK successfully constructed an initial integer feasible solution.
6	MSK_IINF_ANA_PRO_NUM_VAR Number of variables in the problem.
95	MSK_IINF_STO_NUM_A_CACHE_FLUSHES Number of times the cache of A elements is flushed. A large number implies that <code>maxnumanz</code> is too small as well as an inefficient usage of MOSEK.
91	MSK_IINF_SOL_INT_PROSTA Problem status of the integer solution. Updated after each optimization.
66	MSK_IINF_SIM_NETWORK_DUAL_DEG_ITER The number of dual network degenerate iterations.
92	MSK_IINF_SOL_INT_SOLSTA Solution status of the integer solution. Updated after each optimization.
15	MSK_IINF_CACHE_SIZE_L1 L1 cache size used.
16	MSK_IINF_CACHE_SIZE_L2 L2 cache size used.
59	MSK_IINF_RD_NUMVAR Number of variables read.
79	MSK_IINF_SIM_PRIMAL_DUAL_DEG_ITER The number of degenerate major iterations taken by the primal dual simplex algorithm.
12	MSK_IINF_ANA_PRO_NUM_VAR_LO Number of variables with a lower bound and an infinite upper bound.
47	MSK_IINF_MIO_TOTAL_NUM_LIFT_CUTS Number of lift cuts.
89	MSK_IINF_SOL_BAS_PROSTA Problem status of the basic solution. Updated after each optimization.
75	MSK_IINF_SIM_NETWORK_PRIMAL_ITER Number of primal network simplex iterations during the last optimization.
3	MSK_IINF_ANA_PRO_NUM_CON_LO

continued on next page

	continued from previous page
	Number of constraints with a lower bound and an infinite upper bound.
44	MSK_IINF_MIO_TOTAL_NUM_GUB_COVER_CUTS Number of GUB cover cuts.
68	MSK_IINF_SIM_NETWORK_DUAL_HOTSTART_LU If 1 then a valid basis factorization of full rank was located and used by the dual network simplex algorithm.
74	MSK_IINF_SIM_NETWORK_PRIMAL_INF_ITER The number of iterations taken with primal infeasibility in the network optimizer.
84	MSK_IINF_SIM_PRIMAL_HOTSTART If 1 then the primal simplex algorithm is solving from an advanced basis.
26	MSK_IINF_MIO_NUM_BRANCH Number of branches performed during the optimization.
96	MSK_IINF_STO_NUM_A_REALLOC Number of times the storage for storing A has been changed. A large value may indicates that memory fragmentation may occur.
29	MSK_IINF_MIO_NUM_RELAX Number of relaxations solved during the optimization.
34	MSK_IINF_MIO_TOTAL_NUM_BRANCH Number of branches performed during the optimization.
42	MSK_IINF_MIO_TOTAL_NUM_GCD_CUTS Number of gcd cuts.
41	MSK_IINF_MIO_TOTAL_NUM_FLOW_COVER_CUTS Number of flow cover cuts.
28	MSK_IINF_MIO_NUM_INT_SOLUTIONS Number of integer feasible solutions that has been found.
85	MSK_IINF_SIM_PRIMAL_HOTSTART_LU If 1 then a valid basis factorization of full rank was located and used by the primal simplex algorithm.
18	MSK_IINF_CPU_TYPE The type of cpu detected.
1	MSK_IINF_ANA_PRO_NUM_CON_EQ Number of equality constraints.
13	MSK_IINF_ANA_PRO_NUM_VAR_RA Number of variables with finite lower and upper bounds.
86	MSK_IINF_SIM_PRIMAL_INF_ITER The number of iterations taken with primal infeasibility.

continued on next page

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

continued on next page

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

F.18 Information item types

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

F.19 Input/output modes

Value	Name	Description
0	MSK_IOMODE_READ	The file is read-only.
1	MSK_IOMODE_WRITE	The file is write-only. If the file exists then it is truncated when it is opened. Otherwise it is created when it is opened.
2	MSK_IOMODE_READWRITE	The file is to read and written.

F.20 Integer parameters

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

continued on next page

continued from previous page

	Controls the amount of logging from the optimizers employed during the sensitivity analysis. 0 means no logging information is produced.
109	<code>MSK_IPAR_OPF_WRITE_SOL_ITG</code> If <code>MSK_IPAR_OPF_WRITE_SOLUTIONS</code> is <code>MSK_ON</code> and an integer solution is defined, write the integer solution in OPF files.
186	<code>MSK_IPAR_WRITE_BAS_HEAD</code> Controls whether the header section is written to the basic solution file.
79	<code>MSK_IPAR_MIO_BRANCH_PRIORITIES_USE</code> Controls whether branching priorities are used by the mixed-integer optimizer.
83	<code>MSK_IPAR_MIO_CUT_LEVEL_TREE</code> Controls the cut level employed by the mixed-integer optimizer at the tree. See <code>MSK_IPAR_MIO_CUT_LEVEL_ROOT</code> for an explanation of the parameter values.
188	<code>MSK_IPAR_WRITE_DATA_COMPRESSED</code> Controls whether the data file is compressed while it is written. 0 means no compression while higher values mean more compression.
140	<code>MSK_IPAR_READ_MPS_RELAX</code> If this option is turned on, then mixed integer constraints are ignored when a problem is read.
106	<code>MSK_IPAR_OPF_WRITE_PARAMETERS</code> Write a parameter section in an OPF file.
129	<code>MSK_IPAR_READ_CON</code> Expected maximum number of constraints to be read. The option is only used by fast MPS and LP file readers.
196	<code>MSK_IPAR_WRITE_INT_VARIABLES</code> Controls whether the variables section is written to the integer solution file.
123	<code>MSK_IPAR_READ_ADD_ANZ</code> Additional number of non-zeros in A that is made room for in the problem.
36	<code>MSK_IPAR_INTPNT_ORDER_METHOD</code> Controls the ordering strategy used by the interior-point optimizer when factorizing the Newton equation system.
110	<code>MSK_IPAR_OPF_WRITE_SOL_ITR</code> If <code>MSK_IPAR_OPF_WRITE_SOLUTIONS</code> is <code>MSK_ON</code> and an interior solution is defined, write the interior solution in OPF files.
69	<code>MSK_IPAR_LOG_SENSITIVITY</code>

continued on next page

continued from previous page

	Controls the amount of logging during the sensitivity analysis. 0: Means no logging information is produced. 1: Timing information is printed. 2: Sensitivity results are printed.
143	MSK_IPAR_READ_QNZ Expected maximum number of Q non-zeros to be read. The option is used only by MPS and LP file readers.
59	MSK_IPAR_LOG_INFEAS_ANA Controls amount of output printed by the infeasibility analyzer procedures. A higher level implies that more information is logged.
168	MSK_IPAR_SIM_PRIMAL_SELECTION Controls the choice of the incoming variable, known as the selection strategy, in the primal simplex optimizer.
194	MSK_IPAR_WRITE_INT_CONSTRAINTS Controls whether the constraint section is written to the integer solution file.
199	MSK_IPAR_WRITE_LP_STRICT_FORMAT Controls whether LP output files satisfy the LP format strictly.
148	MSK_IPAR_SENSITIVITY_TYPE Controls which type of sensitivity analysis is to be performed.
153	MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION The dual simplex optimizer can use a so-called restricted selection/pricing strategy to chooses the outgoing variable. Hence, if restricted selection is applied, then the dual simplex optimizer first choose a subset of all the potential outgoing variables. Next, for some time it will choose the outgoing variable only among the subset. From time to time the subset is redefined. A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.
62	MSK_IPAR_LOG_MIO_FREQ Controls how frequent the mixed-integer optimizer prints the log line. It will print line every time MSK_IPAR_LOG_MIO_FREQ relaxations have been solved.
108	MSK_IPAR_OPF_WRITE_SOL_BAS If MSK_IPAR_OPF_WRITE_SOLUTIONS is MSK_ON and a basic solution is defined, include the basic solution in OPF files.
14	MSK_IPAR_CHECK_TASK_DATA If this feature is turned on, then the task data is checked for bad values i.e. NaNs. before an optimization is performed.

continued on next page

continued from previous page	
99	MSK_IPAR_MIO_ROOT_OPTIMIZER Controls which optimizer is employed at the root node in the mixed-integer optimizer.
191	MSK_IPAR_WRITE_FREE_CON Controls whether the free constraints are written to the data file.
115	MSK_IPAR_PRESOLVE_ELIM_FILL Controls the maximum amount of fill-in that can be created during the elimination phase of the presolve. This parameter times ($\text{numcon} + \text{numvar}$) denotes the amount of fill-in.
101	MSK_IPAR_NONCONVEX_MAX_ITERATIONS Maximum number of iterations that can be used by the nonconvex optimizer.
88	MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER Controls the size of the local search space when doing local branching.
192	MSK_IPAR_WRITE_GENERIC_NAMES Controls whether the generic names or user-defined names are used in the data file.
184	MSK_IPAR_WARNING_LEVEL Warning level.
51	MSK_IPAR_LOG_BI_FREQ Controls how frequent the optimizer outputs information about the basis identification and how frequent the user-defined call-back function is called.
16	MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX Priority of the dual simplex algorithm when selecting solvers for concurrent optimization.
67	MSK_IPAR_LOG_PRESOLVE Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.
126	MSK_IPAR_READ_ADD_QNZ Additional number of non-zeros in the Q matrices that is made room for in the problem.
206	MSK_IPAR_WRITE_SOL_CONSTRAINTS Controls whether the constraint section is written to the solution file.
35	MSK_IPAR_INTPNT_OFF_COL_TRH

continued on next page

 continued from previous page

	Controls how many offending columns are detected in the Jacobian of the constraint matrix. 1 means aggressive detection, higher values mean less aggressive detection. 0 means no detection.
128	MSK_IPAR_READ_ANZ Expected maximum number of A non-zeros to be read. The option is used only by fast MPS and LP file readers.
92	MSK_IPAR_MIO_MODE Controls whether the optimizer includes the integer restrictions when solving a (mixed) integer optimization problem.
134	MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU If this option is turned on, MOSEK will drop variables that are defined for the first time in the bounds section.
71	MSK_IPAR_LOG_SIM Controls amount of output printed by the simplex optimizer. A higher level implies that more information is logged.
41	MSK_IPAR_LIC_TRH_EXPIRY_WRN If a license feature expires in a numbers days less than the value of this parameter then a warning will be issued.
182	MSK_IPAR_SOLUTION_CALLBACK Indicates whether solution call-backs will be performed during the optimization.
173	MSK_IPAR_SIM_SCALING_METHOD Controls how the problem is scaled before a simplex optimizer is used.
72	MSK_IPAR_LOG_SIM_FREQ Controls how frequent the simplex optimizer outputs information about the optimization and how frequent the user-defined call-back function is called.
63	MSK_IPAR_LOG_NONCONVEX Controls amount of output printed by the nonconvex optimizer.
22	MSK_IPAR_FEASREPAIR_OPTIMIZE Controls which type of feasibility analysis is to be performed.
198	MSK_IPAR_WRITE_LP_QUOTED_NAMES If this option is turned on, then MOSEK will quote invalid LP names when writing an LP file.
55	MSK_IPAR_LOG_FACTOR If turned on, then the factor log lines are added to the log.
4	MSK_IPAR_AUTO_UPDATE_SOL_INFO

continued on next page

continued from previous page	
	Controls whether the solution information items are automatically updated after an optimization is performed.
203	MSK_IPAR_WRITE_MPS_QUOTED_NAMES If a name contains spaces (blanks) when writing an MPS file, then the quotes will be removed.
141	MSK_IPAR_READ_MPS_WIDTH Controls the maximal number of characters allowed in one line of the MPS file.
183	MSK_IPAR_TIMING_LEVEL Controls the amount of timing performed inside MOSEK.
65	MSK_IPAR_LOG_ORDER If turned on, then factor lines are added to the log.
82	MSK_IPAR_MIO_CUT_LEVEL_ROOT Controls the cut level employed by the mixed-integer optimizer at the root node. A negative value means a default value determined by the mixed-integer optimizer is used. By adding the appropriate values from the following table the employed cut types can be controlled.
	GUB cover +2
	Flow cover +4
	Lifting +8
	Plant location +16
	Disaggregation +32
	Knapsack cover +64
	Lattice +128
	Gomory +256
	Coefficient reduction +512
	GCD +1024
	Obj. integrality +2048
5	MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE If a slack variable is in the basis, then the corresponding column in the basis is a unit vector with -1 in the right position. However, if this parameter is set to MSK_ON , -1 is replaced by 1.
8	MSK_IPAR_BI_IGNORE_NUM_ERROR If the parameter MSK_IPAR_INTPNT_BASIS has the value MSK_BI_NO_ERROR and the interior-point optimizer has terminated due to a numerical problem, then basis identification is performed if this parameter has the value MSK_ON .
94	MSK_IPAR_MIO_NODE_SELECTION
continued on next page	

continued from previous page	
	Controls the node selection strategy employed by the mixed-integer optimizer.
2	MSK_IPAR_ANA_SOL_PRINT_VIOLATED
181	MSK_IPAR_SOL_READ_WIDTH
	Controls the maximal acceptable width of line in the solutions when read by MOSEK.
120	MSK_IPAR_PRESOLVE_LINDEP_WORK_LIM
	Is used to limit the amount of work that can be done to locate linear dependencies. In general the higher value this parameter is given the less work can be used. However, a value of 0 means no limit on the amount of work that can be used.
33	MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS
	Maximum number of steps to be used by the iterative refinement of the search direction. A negative value implies that the optimizer chooses the maximum number of iterative refinement steps.
124	MSK_IPAR_READ_ADD_CON
	Additional number of constraints that is made room for in the problem.
53	MSK_IPAR_LOG_CONCURRENT
	Controls amount of output printed by the concurrent optimizer.
73	MSK_IPAR_LOG_SIM_MINOR
	Currently not in use.
159	MSK_IPAR_SIM_MAX_ITERATIONS
	Maximum number of iterations that can be used by a simplex optimizer.
31	MSK_IPAR_INTPNT_MAX_ITERATIONS
	Controls the maximum number of iterations allowed in the interior-point optimizer.
20	MSK_IPAR_CPU_TYPE
	Specifies the CPU type. By default MOSEK tries to auto detect the CPU type. Therefore, we recommend to change this parameter only if the auto detection does not work properly.
50	MSK_IPAR_LOG_BI
	Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.
32	MSK_IPAR_INTPNT_MAX_NUM_COR

continued on next page

continued from previous page

		Controls the maximum number of correctors allowed by the multiple corrector procedure. A negative value means that MOSEK is making the choice.
197	<code>MSK_IPAR_WRITE_LP_LINE_WIDTH</code>	Maximum width of line in an LP file written by MOSEK.
180	<code>MSK_IPAR_SOL_READ_NAME_WIDTH</code>	When a solution is read by MOSEK and some constraint, variable or cone names contain blanks, then a maximum name width much be specified. A negative value implies that no name contain blanks.
45	<code>MSK_IPAR_LICENSE_DEBUG</code>	This option is used to turn on debugging of the incense manager.
48	<code>MSK_IPAR_LICENSE_WAIT</code>	If all licenses are in use MOSEK returns with an error code. However, by turning on this parameter MOSEK will wait for an available license.
1	<code>MSK_IPAR_ANA_SOL_BASIS</code>	Controls whether the basis matrix is analyzed in solaution analyzer.
116	<code>MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES</code>	Control the maximum number of times the eliminator is tried.
193	<code>MSK_IPAR_WRITE_GENERIC_NAMES_IO</code>	Index origin used in generic names.
15	<code>MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS</code>	The maximum number of simultaneous optimizations that will be started by the concurrent optimizer.
169	<code>MSK_IPAR_SIM_REFACTOR_FREQ</code>	Controls how frequent the basis is refactorized. The value 0 means that the optimizer determines the best point of refactorization. It is strongly recommended NOT to change this parameter.
154	<code>MSK_IPAR_SIM_DUAL_SELECTION</code>	Controls the choice of the incoming variable, known as the selection strategy, in the dual simplex optimizer.
174	<code>MSK_IPAR_SIM_SOLVE_FORM</code>	Controls whether the primal or the dual problem is solved by the primal-/dual- simplex optimizer.
13	<code>MSK_IPAR_CHECK_CONVEXITY</code>	Specify the level of convexity check on quadratic problems
122	<code>MSK_IPAR_QO_SEPARABLE_REFORMULATION</code>	Determine if Quadratic programing problems should be reformulated to separable form.

continued on next page

 continued from previous page

76	<code>MSK_IPAR_LP_WRITE_IGNORE_INCOMPATIBLE_ITEMS</code>	Controls the result of writing a problem containing incompatible items to an LP file.
144	<code>MSK_IPAR_READ_TASK_IGNORE_PARAM</code>	Controls whether MOSEK should ignore the parameter setting defined in the task file and use the default parameter setting instead.
95	<code>MSK_IPAR_MIO_OPTIMIZER_MODE</code>	An experimental feature.
60	<code>MSK_IPAR_LOG_INTPNT</code>	Controls amount of output printed by the interior-point optimizer. A higher level implies that more information is logged.
61	<code>MSK_IPAR_LOG_MIO</code>	Controls the log level for the mixed-integer optimizer. A higher level implies that more information is logged.
156	<code>MSK_IPAR_SIM_HOTSTART</code>	Controls the type of hot-start that the simplex optimizer perform.
66	<code>MSK_IPAR_LOG_PARAM</code>	Controls the amount of information printed out about parameter changes.
189	<code>MSK_IPAR_WRITE_DATA_FORMAT</code>	Controls the file format when writing task data to a file.
155	<code>MSK_IPAR_SIM_EXPLOIT_DUPVEC</code>	Controls if the simplex optimizers are allowed to exploit duplicated columns.
78	<code>MSK_IPAR_MIO_BRANCH_DIR</code>	Controls whether the mixed-integer optimizer is branching up or down by default.
29	<code>MSK_IPAR_INTPNT_FACTOR_DEBUG_LVL</code>	Controls factorization debug level.
179	<code>MSK_IPAR_SOL_QUOTED_NAMES</code>	If this options is turned on, then MOSEK will quote names that contains blanks while writing the solution file. Moreover when reading leading and trailing quotes will be stripped of.
46	<code>MSK_IPAR_LICENSE_PAUSE_TIME</code>	If <code>MSK_IPAR_LICENSE_WAIT=MSK_ON</code> and no license is available, then MOSEK sleeps a number of milliseconds between each check of whether a license has become free.
96	<code>MSK_IPAR_MIO_PRESOLVE_AGGREGATE</code>	

continued on next page

continued from previous page	
	Controls whether the presolve used by the mixed-integer optimizer tries to aggregate the constraints.
209	<code>MSK_IPAR_WRITE_TASK_INC_SOL</code> Controls whether the solutions are stored in the task file too.
43	<code>MSK_IPAR_LICENSE_CACHE_TIME</code> Setting this parameter no longer has any effect. Please see <code>MSK_IPAR_CACHE_LICENSE</code> for an alternative.
9	<code>MSK_IPAR_BI_MAX_ITERATIONS</code> Controls the maximum number of simplex iterations allowed to optimize a basis after the basis identification.
157	<code>MSK_IPAR_SIM_HOTSTART_LU</code> Determines if the simplex optimizer should exploit the initial factorization.
111	<code>MSK_IPAR_OPF_WRITE_SOLUTIONS</code> Enable inclusion of solutions in the OPF files.
162	<code>MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART</code> This parameter controls how large the network component in “relative” terms has to be before it is exploited in a simplex hot-start. The network component should be equal or larger than <code>max(MSK_IPAR_SIM_NETWORK_DETECT, MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART)</code> before it is exploited. If this value is larger than 100 the network flow component is never detected or exploited.
119	<code>MSK_IPAR_PRESOLVE_LINDEP_USE</code> Controls whether the linear constraints are checked for linear dependencies.
114	<code>MSK_IPAR_PARAM_READ_IGN_ERROR</code> If turned on, then errors in parameter settings is ignored.
104	<code>MSK_IPAR_OPF_WRITE_HEADER</code> Write a text header with date and MOSEK version in an OPF file.
81	<code>MSK_IPAR_MIO_CONT_SOL</code> Controls the meaning of the interior-point and basic solutions in mixed integer problems.
102	<code>MSK_IPAR_OBJECTIVE_SENSE</code> If the objective sense for the task is undefined, then the value of this parameter is used as the default objective sense.
195	<code>MSK_IPAR_WRITE_INT_HEAD</code>

continued on next page

continued from previous page

	Controls whether the header section is written to the integer solution file.
40	MSK_IPAR_INTPNT_STARTING_POINT Starting point used by the interior-point optimizer.
49	MSK_IPAR_LOG Controls the amount of log information. The value 0 implies that all log information is suppressed. A higher level implies that more information is logged. Please note that if a task is employed to solve a sequence of optimization problems the value of this parameter is reduced by the value of MSK_IPAR_LOG_CUT_SECOND_OPT for the second and any subsequent optimizations.
19	MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX Priority of the primal simplex algorithm when selecting solvers for concurrent optimization.
138	MSK_IPAR_READ_MPS_OBJ_SENSE If turned on, the MPS reader uses the objective sense section. Otherwise the MPS reader ignores it.
10	MSK_IPAR_CACHE_LICENSE Specifies if the license is kept checked out for the lifetime of the mosek environment (on) or returned to the server immediately after the optimization (off). Check-in and check-out of licenses have an overhead. Frequent communication with the license server should be avoided.
74	MSK_IPAR_LOG_SIM_NETWORK_FREQ Controls how frequent the network simplex optimizer outputs information about the optimization and how frequent the user-defined call-back function is called. The network optimizer will use a logging frequency equal to MSK_IPAR_LOG_SIM_FREQ times MSK_IPAR_LOG_SIM_NETWORK_FREQ .
28	MSK_IPAR_INTPNT_DIFF_STEP Controls whether different step sizes are allowed in the primal and dual space.
172	MSK_IPAR_SIM_SCALING Controls how much effort is used in scaling the problem before a simplex optimizer is used.
200	MSK_IPAR_WRITE_LP_TERMS_PER_LINE Maximum number of terms on a single line in an LP file written by MOSEK. 0 means unlimited.

continued on next page

continued from previous page	
146	MSK_IPAR_SENSITIVITY_ALL Not applicable.
178	MSK_IPAR_SOL_FILTER_KEEP_RANGED If turned on, then ranged constraints and variables are written to the solution file independent of the filter setting.
7	MSK_IPAR_BI_IGNORE_MAX_ITER If the parameter MSK_IPAR_INTPNT_BASIS has the value MSK_BI_NO_ERROR and the interior-point optimizer has terminated due to maximum number of iterations, then basis identification is performed if this parameter has the value MSK_ON .
56	MSK_IPAR_LOG_FEASREPAIR Controls the amount of output printed when performing feasibility repair.
39	MSK_IPAR_INTPNT_SOLVE_FORM Controls whether the primal or the dual problem is solved.
103	MSK_IPAR_OPF_MAX_TERMS_PER_LINE The maximum number of terms (linear and quadratic) per line when an OPF file is written.
205	MSK_IPAR_WRITE_PRECISION Controls the precision with which double numbers are printed in the MPS data file. In general it is not worthwhile to use a value higher than 15.
149	MSK_IPAR_SIM_BASIS_FACTOR_USE 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 penalty.
210	MSK_IPAR_WRITE_XML_MODE Controls if linear coefficients should be written by row or column when writing in the XML file format.
37	MSK_IPAR_INTPNT_REGULARIZATION_USE Controls whether regularization is allowed.
6	MSK_IPAR_BI_CLEAN_OPTIMIZER Controls which simplex optimizer is used in the clean-up phase.
97	MSK_IPAR_MIO_PRESOLVE_PROBING Controls whether the mixed-integer presolve performs probing. Probing can be very time consuming.
42	MSK_IPAR_LICENSE_ALLOW_OVERUSE

continued on next page

	continued from previous page
	Controls if license overuse is allowed when caching licenses
24	MSK_IPAR_INFEAS_PREFER_PRIMAL If both certificates of primal and dual infeasibility are supplied then only the primal is used when this option is turned on.
187	MSK_IPAR_WRITE_BAS_VARIABLES Controls whether the variables section is written to the basic solution file.
75	MSK_IPAR_LOG_STORAGE When turned on, MOSEK prints messages regarding the storage usage and allocation.
98	MSK_IPAR_MIO_PRESOLVE_USE Controls whether presolve is performed by the mixed-integer optimizer.
135	MSK_IPAR_READ_LP_QUOTED_NAMES If a name is in quotes when reading an LP file, the quotes will be removed.
27	MSK_IPAR_INTPNT_BASIS Controls whether the interior-point optimizer also computes an optimal basis.
54	MSK_IPAR_LOG_CUT_SECOND_OPT If a task is employed to solve a sequence of optimization problems, then the value of the log levels is reduced by the value of this parameter. E.g MSK_IPAR_LOG and MSK_IPAR_LOG_SIM are reduced by the value of this parameter for the second and any subsequent optimizations.
137	MSK_IPAR_READ_MPS_KEEP_INT Controls whether MOSEK should keep the integer restrictions on the variables while reading the MPS file.
91	MSK_IPAR_MIO_MAX_NUM_SOLUTIONS The mixed-integer optimizer can be terminated after a certain number of different feasible solutions has been located. If this parameter has the value n and n is strictly positive, then the mixed-integer optimizer will be terminated when n feasible solutions have been located.
44	MSK_IPAR_LICENSE_CHECK_TIME The parameter specifies the number of seconds between the checks of all the active licenses in the MOSEK environment license cache. These checks are performed to determine if the licenses should be returned to the server.
208	MSK_IPAR_WRITE_SOL_VARIABLES

continued on next page

continued from previous page	
	Controls whether the variables section is written to the solution file.
147	MSK_IPAR_SENSITIVITY_OPTIMIZER Controls which optimizer is used for optimal partition sensitivity analysis.
201	MSK_IPAR_WRITE_MPS_INT Controls if marker records are written to the MPS file to indicate whether variables are integer restricted.
160	MSK_IPAR_SIM_MAX_NUM_SETBACKS Controls how many set-backs are allowed within a simplex optimizer. A set-back is an event where the optimizer moves in the wrong direction. This is impossible in theory but may happen due to numerical problems.
21	MSK_IPAR_DATA_CHECK If this option is turned on, then extensive data checking is enabled. It will slow down MOSEK but on the other hand help locating bugs.
17	MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX Priority of the free simplex optimizer when selecting solvers for concurrent optimization.
133	MSK_IPAR_READ_KEEP_FREE_CON Controls whether the free constraints are included in the problem.
57	MSK_IPAR_LOG_FILE If turned on, then some log info is printed when a file is written or read.
18	MSK_IPAR_CONCURRENT_PRIORITY_INTPNT Priority of the interior-point algorithm when selecting solvers for concurrent optimization.
164	MSK_IPAR_SIM_NON_SINGULAR Controls if the simplex optimizer ensures a non-singular basis, if possible.
190	MSK_IPAR_WRITE_DATA_PARAM If this option is turned on the parameter settings are written to the data file as parameters.
150	MSK_IPAR_SIM_DEGEN Controls how aggressively degeneration is handled.
105	MSK_IPAR_OPF_WRITE_HINTS Write a hint section with problem dimensions in the beginning of an OPF file.
117	MSK_IPAR_PRESOLVE_ELIMINATOR_USE

continued on next page

continued from previous page	
	Controls whether free or implied free variables are eliminated from the problem.
0	MSK_IPAR_ALLOC_ADD_QNZ Additional number of Q non-zeros that are allocated space for when numanz exceeds maxnumqnz during addition of new Q entries.
86	MSK_IPAR_MIO_HOTSTART Controls whether the integer optimizer is hot-started.
136	MSK_IPAR_READ_MPS_FORMAT Controls how strictly the MPS file reader interprets the MPS format.
113	MSK_IPAR_PARAM_READ_CASE_NAME If turned on, then names in the parameter file are case sensitive.
139	MSK_IPAR_READ_MPS_QUOTED_NAMES If a name is in quotes when reading an MPS file, then the quotes will be removed.
64	MSK_IPAR_LOG_OPTIMIZER Controls the amount of general optimizer information that is logged.
202	MSK_IPAR_WRITE_MPS_OBJ_SENSE If turned off, the objective sense section is not written to the MPS file.
34	MSK_IPAR_INTPNT_NUM_THREADS Controls the number of threads employed by the interior-point optimizer. If set to a positive number MOSEK will use this number of threads. If zero the number of threads used will equal the number of cores detected on the machine.
89	MSK_IPAR_MIO_MAX_NUM_BRANCHES Maximum number of branches allowed during the branch and bound search. A negative value means infinite.
165	MSK_IPAR_SIM_PRIMAL_CRASH Controls whether crashing is performed in the primal simplex optimizer. In general, if a basis consists of more than $(100 - \text{this parameter value})\%$ fixed variables, then a crash will be performed.
80	MSK_IPAR_MIO_CONSTRUCT_SOL If set to MSK_ON and all integer variables have been given a value for which a feasible mixed integer solution exists, then MOSEK generates an initial solution to the mixed integer problem by fixing all integer values and solving the remaining problem.
3	MSK_IPAR_AUTO_SORT_A_BEFORE_OPT

continued on next page

continued from previous page	
	Controls whether the elements in each column of A are sorted before an optimization is performed. This is not required but makes the optimization more deterministic.
100	MSK_IPAR_MIO_STRONG_BRANCH The value specifies the depth from the root in which strong branching is used. A negative value means that the optimizer chooses a default value automatically.
152	MSK_IPAR_SIM_DUAL_PHASEONE_METHOD An experimental feature.
158	MSK_IPAR_SIM_INTEGER An experimental feature.
167	MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION The primal simplex optimizer can use a so-called restricted selection/pricing strategy to chooses the outgoing variable. Hence, if restricted selection is applied, then the primal simplex optimizer first choose a subset of all the potential incoming variables. Next, for some time it will choose the incoming variable only among the subset. From time to time the subset is redefined. A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.
130	MSK_IPAR_READ_CONE Expected maximum number of conic constraints to be read. The option is used only by fast MPS and LP file readers.
112	MSK_IPAR_OPTIMIZER The paramter controls which optimizer is used to optimize the task.
77	MSK_IPAR_MAX_NUM_WARNINGS Waning level. A higher value results in more warnings.
47	MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS Controls whether license features expire warnings are suppressed.
207	MSK_IPAR_WRITE_SOL_HEAD Controls whether the header section is written to the solution file.
185	MSK_IPAR_WRITE_BAS_CONSTRAINTS Controls whether the constraint section is written to the basic solution file.
84	MSK_IPAR_MIO_FEASPUMP_LEVEL

continued on next page

 continued from previous page

- Feasibility pump is a heuristic designed to compute an initial feasible solution. A value of 0 implies that the feasibility pump heuristic is not used. A value of -1 implies that the mixed-integer optimizer decides how the feasibility pump heuristic is used. A larger value than 1 implies that the feasibility pump is employed more aggressively. Normally a value beyond 3 is not worthwhile.
- 23 **MSK_IPAR_INFEAS_GENERIC_NAMES**
Controls whether generic names are used when an infeasible subproblem is created.
- 161 **MSK_IPAR_SIM_NETWORK_DETECT**
The simplex optimizer is capable of exploiting a network flow component in a problem. However it is only worthwhile to exploit the network flow component if it is sufficiently large. This parameter controls how large the network component has to be in “relative” terms before it is exploited. For instance a value of 20 means at least 20% of the model should be a network before it is exploited. If this value is larger than 100 the network flow component is never detected or exploited.
- 68 **MSK_IPAR_LOG_RESPONSE**
Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.
- 26 **MSK_IPAR_INFEAS_REPORT_LEVEL**
Controls the amount of information presented in an infeasibility report. Higher values imply more information.
- 11 **MSK_IPAR_CACHE_SIZE_L1**
Specifies the size of the cache of the computer. This parameter is potentially very important for the efficiency on computers if MOSEK cannot determine the cache size automatically. If the cache size is negative, then MOSEK tries to determine the value automatically.
- 12 **MSK_IPAR_CACHE_SIZE_L2**
Specifies the size of the cache of the computer. This parameter is potentially very important for the efficiency on computers where MOSEK cannot determine the cache size automatically. If the cache size is negative, then MOSEK tries to determine the value automatically.
- 176 **MSK_IPAR_SIM_SWITCH_OPTIMIZER**
-

continued on next page

continued from previous page

	The simplex optimizer sometimes chooses to solve the dual problem instead of the primal problem. This implies that if you have chosen to use the dual simplex optimizer and the problem is dualized, then it actually makes sense to use the primal simplex optimizer instead. If this parameter is on and the problem is dualized and furthermore the simplex optimizer is chosen to be the primal (dual) one, then it is switched to the dual (primal).
131	MSK_IPAR_READ_DATA_COMPRESSED If this option is turned on, it is assumed that the data file is compressed.
142	MSK_IPAR_READ_Q_MODE Controls how the Q matrices are read from the MPS file.
107	MSK_IPAR_OPF_WRITE_PROBLEM Write objective, constraints, bounds etc. to an OPF file.
52	MSK_IPAR_LOG_CHECK_CONVEXITY Controls logging in convexity check on quadratic problems. Set to a positive value to turn logging on. If a quadratic coefficient matrix is found to violate the requirement of PSD (NSD) then a list of negative (positive) pivot elements is printed. The absolute value of the pivot elements is also shown.
132	MSK_IPAR_READ_DATA_FORMAT Format of the data file to be read.
151	MSK_IPAR_SIM_DUAL_CRASH Controls whether crashing is performed in the dual simplex optimizer. In general if a basis consists of more than (100-this parameter value)% fixed variables, then a crash will be performed.
163	MSK_IPAR_SIM_NETWORK_DETECT_METHOD Controls which type of detection method the network extraction should use.
145	MSK_IPAR_READ_VAR Expected maximum number of variable to be read. The option is used only by MPS and LP file readers.
58	MSK_IPAR_LOG_HEAD If turned on, then a header line is added to the log.
170	MSK_IPAR_SIM_REFORMULATION Controls if the simplex optimizers are allowed to reformulate the problem.
171	MSK_IPAR_SIM_SAVE_LU

continued on next page

continued from previous page	
	Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.
30	MSK_IPAR_INTPNT_FACTOR_METHOD Controls the method used to factor the Newton equation system.
90	MSK_IPAR_MIO_MAX_NUM_RELAXS Maximum number of relaxations allowed during the branch and bound search. A negative value means infinite.
177	MSK_IPAR_SOL_FILTER_KEEP_BASIC If turned on, then basic and super basic constraints and variables are written to the solution file independent of the filter setting.
85	MSK_IPAR_MIO_HEURISTIC_LEVEL Controls the heuristic employed by the mixed-integer optimizer to locate an initial good integer feasible solution. A value of zero means the heuristic is not used at all. A larger value than 0 means that a gradually more sophisticated heuristic is used which is computationally more expensive. A negative value implies that the optimizer chooses the heuristic. Normally a value around 3 to 5 should be optimal.
87	MSK_IPAR_MIO_KEEP_BASIS Controls whether the integer presolve keeps bases in memory. This speeds on the solution process at cost of bigger memory consumption.
38	MSK_IPAR_INTPNT_SCALING Controls how the problem is scaled before the interior-point optimizer is used.

F.21 Language selection constants

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

F.22 Long integer information items.

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

F.23 Mark

Value	Name	Description
0	MSK_MARK_LO	The lower bound is selected for sensitivity analysis.
1	MSK_MARK_UP	The upper bound is selected for sensitivity analysis.

F.24 Continuous mixed-integer solution type

Value	Name	Description
2	MSK_MIO_CONT_SOL_ITG	The reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. A solution is only reported in case the problem has a primal feasible solution.
0	MSK_MIO_CONT_SOL_NONE	No interior-point or basic solution are reported when the mixed-integer optimizer is used.
1	MSK_MIO_CONT_SOL_ROOT	The reported interior-point and basic solutions are a solution to the root node problem when mixed-integer optimizer is used.
3	MSK_MIO_CONT_SOL_ITG_REL	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.

F.25 Integer restrictions

Value	Name	Description
0	MSK_MIO_MODE_IGNORED	The integer constraints are ignored and the problem is solved as a continuous problem.
2	MSK_MIO_MODE_LAZY	

continued on next page

	continued from previous page
	Integer restrictions should be satisfied if an optimizer is available for the problem.
1	MSK_MIO_MODE_SATISFIED Integer restrictions should be satisfied.

F.26 Mixed-integer node selection types

Value	Name Description
5	MSK_MIO_NODE_SELECTION_PSEUDO The optimizer employs selects the node based on a pseudo cost estimate.
4	MSK_MIO_NODE_SELECTION_HYBRID The optimizer employs a hybrid strategy.
0	MSK_MIO_NODE_SELECTION_FREE The optimizer decides the node selection strategy.
3	MSK_MIO_NODE_SELECTION_WORST The optimizer employs a worst bound node selection strategy.
2	MSK_MIO_NODE_SELECTION_BEST The optimizer employs a best bound node selection strategy.
1	MSK_MIO_NODE_SELECTION_FIRST The optimizer employs a depth first node selection strategy.

F.27 MPS file format type

Value	Name Description
0	MSK_MPS_FORMAT_STRICT It is assumed that the input file satisfies the MPS format strictly.
1	MSK_MPS_FORMAT_RELAXED It is assumed that the input file satisfies a slightly relaxed version of the MPS format.
2	MSK_MPS_FORMAT_FREE It is assumed that the input file satisfies the free MPS format. This implies that spaces are not allowed in names. Otherwise the format is free.

F.28 Message keys

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

F.29 Network detection method

Value	Name	Description
1	MSK_NETWORK_DETECT_SIMPLE	The network detection should use a very simple heuristic.
2	MSK_NETWORK_DETECT_ADVANCED	The network detection should use a more advanced heuristic.
0	MSK_NETWORK_DETECT_FREE	The network detection is free.

F.30 Objective sense types

Value	Name	Description
1	MSK_OBJECTIVE_SENSE_MINIMIZE	The problem should be minimized.
0	MSK_OBJECTIVE_SENSE_UNDEFINED	The objective sense is undefined.
2	MSK_OBJECTIVE_SENSE_MAXIMIZE	The problem should be maximized.

F.31 On/off

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

F.32 Optimizer types

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

F.33 Ordering strategies

Value	Name	Description
5	MSK_ORDER_METHOD_NONE	No ordering is used.
2	MSK_ORDER_METHOD_APPMINLOC2	A variant of the approximate minimum local-fill-in ordering is used.
1	MSK_ORDER_METHOD_APPMINLOC1	Approximate minimum local-fill-in ordering is used.
4	MSK_ORDER_METHOD_GRAPHPAR2	An alternative graph partitioning based ordering.
0	MSK_ORDER_METHOD_FREE	The ordering method is chosen automatically.
3	MSK_ORDER_METHOD_GRAPHPAR1	Graph partitioning based ordering.

F.34 Parameter type

Value	Name	Description
0	MSK_PAR_INVALID_TYPE	Not a valid parameter.
3	MSK_PAR_STR_TYPE	Is a string parameter.
1	MSK_PAR_DOU_TYPE	Is a double parameter.
2	MSK_PAR_INT_TYPE	Is an integer parameter.

F.35 Presolve method.

Value	Name	Description
1	MSK_PRESOLVE_MODE_ON	The problem is presolved before it is optimized.
0	MSK_PRESOLVE_MODE_OFF	The problem is not presolved before it is optimized.
2	MSK_PRESOLVE_MODE_FREE	

continued on next page

continued from previous page

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

F.36 Problem data items

Value	Name	Description
0	MSK_PI_VAR	Item is a variable.
2	MSK_PI_CONE	Item is a cone.
1	MSK_PI_CON	Item is a constraint.

F.37 Problem types

Value	Name	Description
2	MSK_PROBTYPE_QCQO	The problem is a quadratically constrained optimization problem.
0	MSK_PROBTYPE_LO	The problem is a linear optimization problem.
4	MSK_PROBTYPE_CONIC	A conic optimization.
3	MSK_PROBTYPE_GECO	General convex optimization.
5	MSK_PROBTYPE_MIXED	General nonlinear constraints and conic constraints. This combination can not be solved by MOSEK.
1	MSK_PROBTYPE_QO	The problem is a quadratic optimization problem.

F.38 Problem status keys

Value	Name	Description
6	MSK_PRO_STA_PRIM_AND_DUAL_INFEAS	The problem is primal and dual infeasible.
4	MSK_PRO_STA_PRIM_INFEAS	The problem is primal infeasible.
7	MSK_PRO_STA_ILL_POSED	The problem is ill-posed. For example, it may be primal and dual feasible but have a positive duality gap.
0	MSK_PRO_STA_UNKNOWN	Unknown problem status.
2	MSK_PRO_STA_PRIM_FEAS	The problem is primal feasible.
8	MSK_PRO_STA_NEAR_PRIM_AND_DUAL_FEAS	The problem is at least nearly primal and dual feasible.
10	MSK_PRO_STA_NEAR_DUAL_FEAS	The problem is at least nearly dual feasible.
11	MSK_PRO_STA_PRIM_INFEAS_OR_UNBOUNDED	The problem is either primal infeasible or unbounded. This may occur for mixed-integer problems.
1	MSK_PRO_STA_PRIM_AND_DUAL_FEAS	The problem is primal and dual feasible.
5	MSK_PRO_STA_DUAL_INFEAS	The problem is dual infeasible.
9	MSK_PRO_STA_NEAR_PRIM_FEAS	The problem is at least nearly primal feasible.
3	MSK_PRO_STA_DUAL_FEAS	The problem is dual feasible.

F.39 Interpretation of quadratic terms in MPS files

Value	Name	Description
0	MSK_Q_READ_ADD	All elements in a Q matrix are assumed to belong to the lower triangular part. Duplicate elements in a Q matrix are added together.
1	MSK_Q_READ_DROP_LOWER	All elements in the strict lower triangular part of the Q matrices are dropped.

continued on next page

continued from previous page	
2	MSK_Q_READ_DROP_UPPER
	All elements in the strict upper triangular part of the Q matrices are dropped.

F.40 Response codes

Value	Name	Description
352	MSK_RES_WRN_SOL_FILE_IGNORED_VAR	One or more lines in the variable section were ignored when reading a solution file.
1218	MSK_RES_ERR_PARAM_TYPE	The parameter type is invalid.
1203	MSK_RES_ERR_INDEX_IS_TOO_SMALL	An index in an argument is too small.
2501	MSK_RES_ERR_INV_MARKI	Invalid value in marki.
803	MSK_RES_WRN_PRESOLVE_BAD_PRECISION	The presolve estimates that the model is specified with insufficient precision.
1500	MSK_RES_ERR_INV_PROBLEM	Invalid problem type. Probably a nonconvex problem has been specified.
1268	MSK_RES_ERR_INV_SKX	Invalid value in <code>skx</code> .
1551	MSK_RES_ERR_MIO_NO_OPTIMIZER	No optimizer is available for the current class of integer optimization problems.
4009	MSK_RES_TRM_MIO_NUM_BRANCHES	The mixed-integer optimizer terminated as to the maximum number of branches was reached.
4004	MSK_RES_TRM_MIO_NEAR_ABS_GAP	The mixed-integer optimizer terminated because the near optimal absolute gap tolerance was satisfied.
2001	MSK_RES_ERR_NO_DUAL_INFEAS_CER	A certificate of infeasibility is not available.
1254	MSK_RES_ERR_MUL_A_ELEMENT	An element in A is defined multiple times.

continued on next page

 continued from previous page

1170	MSK_RES_ERR_INVALID_NAME_IN_SOL_FILE	An invalid name occurred in a solution file.
1114	MSK_RES_ERR_MPS_MUL_QOBJ	The Q term in the objective is specified multiple times in the MPS data file.
1063	MSK_RES_ERR_NO_INIT_ENV	env is not initialized.
1265	MSK_RES_ERR_UNDEF_SOLUTION	MOSEK has the following solution types: <ul style="list-style-type: none"> • an interior-point solution, • an basic solution, • and an integer solution. <p>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.</p>
1288	MSK_RES_ERR_LASTJ	Invalid lastj.
1001	MSK_RES_ERR_LICENSE_EXPIRED	The license has expired.
3055	MSK_RES_ERR_SEN_INDEX_INVALID	Invalid range given in the sensitivity file.
1274	MSK_RES_ERR_INV_SKN	Invalid value in skn.
1295	MSK_RES_ERR_OBJ_Q_NOT_PSD	The quadratic coefficient matrix in the objective is not positive semi-definite as expected for a minimization problem.
1234	MSK_RES_ERR_INF_LINT_NAME	A long integer information name is invalid.
903	MSK_RES_WRN_ANA_CLOSE_BOUNDS	This warning is issued by problem analyzer, if ranged constraints or variables with very close upper and lower bounds are detected. One should consider treating such constraints as equalities and such variables as constants.
1008	MSK_RES_ERR_MISSING_LICENSE_FILE	

 continued on next page

continued from previous page	
	MOSEK cannot find the license file or license server. Usually this happens if the operating system variable MOSEKLM.LICENSE.FILE is not set up appropriately. Please see the MOSEK installation manual for details.
1235	MSK_RES_ERR_INDEX An index is out of range.
1350	MSK_RES_ERR_SOL_FILE_INVALID_NUMBER An invalid number is specified in a solution file.
2800	MSK_RES_ERR_LU_MAX_NUM_TRIES Could not compute the LU factors of the matrix within the maximum number of allowed tries.
1267	MSK_RES_ERR_INV_SKC Invalid value in <code>skc</code> .
201	MSK_RES_WRN_DROPPED_NZ_QOBJ One or more non-zero elements were dropped in the Q matrix in the objective.
3000	MSK_RES_ERR_INTERNAL An internal error occurred. Please report this problem.
1610	MSK_RES_ERR_BASIS_FACTOR The factorization of the basis is invalid.
1204	MSK_RES_ERR_INDEX_IS_TOO_LARGE An index in an argument is too large.
1154	MSK_RES_ERR_LP_INVALID_VAR_NAME A variable name is invalid when used in an LP formatted file.
2950	MSK_RES_ERR_NO_DUAL_FOR_ITG_SOL No dual information is available for the integer solution.
1590	MSK_RES_ERR_OVERFLOW A computation produced an overflow i.e. a very large number.
1150	MSK_RES_ERR_LP_INCOMPATIBLE The problem cannot be written to an LP formatted file.
1501	MSK_RES_ERR_MIXED_PROBLEM The problem contains both conic and nonlinear constraints.
1700	MSK_RES_ERR_FEASREPAIR_CANNOT_RELAX An optimization problem cannot be relaxed. This is the case e.g. for general nonlinear optimization problems.
1207	MSK_RES_ERR_PARAM_NAME_INT The parameter name is not correct for an integer parameter.
3057	MSK_RES_ERR_SEN_SOLUTION_STATUS

continued on next page

	continued from previous page
	No optimal solution found to the original problem given for sensitivity analysis.
1225	MSK_RES_ERR_INF_LINT_INDEX A long integer information index is out of range for the specified type.
4008	MSK_RES_TRM_MIO_NUM_RELAXS The mixed-integer optimizer terminated as the maximum number of relaxations was reached.
405	MSK_RES_WRN_TOO_MANY_BASIS_VARS A basis with too many variables has been specified.
1081	MSK_RES_ERR_SPACE_NO_INFO No available information about the space usage.
1205	MSK_RES_ERR_PARAM_NAME The parameter name is not correct.
1106	MSK_RES_ERR_MPS_UNDEF_VAR_NAME An undefined variable name occurred in an MPS file.
200	MSK_RES_WRN_NZ_IN_UPR_TRI Non-zero elements specified in the upper triangle of a matrix were ignored.
505	MSK_RES_WRN_LICENSE_FEATURE_EXPIRE The license expires.
1263	MSK_RES_ERR_NEGATIVE_SURPLUS Negative surplus.
1404	MSK_RES_ERR_INV_QCON_SUBK Invalid value in <code>qcsbk</code> .
1406	MSK_RES_ERR_INV_QCON_SUBJ Invalid value in <code>qcsubj</code> .
705	MSK_RES_WRN_ZEROS_IN_SPARSE_ROW One or more (near) zero elements are specified in a sparse row of a matrix. It is redundant to specify zero elements. Hence it may indicate an error.
1198	MSK_RES_ERR_ARGUMENT_TYPE Incorrect argument type.
1017	MSK_RES_ERR_LICENSE_MOSEKLM_DAEMON The MOSEKLM license manager daemon is not up and running.
2901	MSK_RES_ERR_INVALID_WCHAR An invalid <code>wchar</code> string is encountered.
1059	MSK_RES_ERR_END_OF_FILE End of file reached.
3102	MSK_RES_ERR_AD_INVALID_CODELIST

continued on next page

continued from previous page	
	The code list data was invalid.
1462	MSK_RES_ERR_NAN_IN_BUC u^c contains an invalid floating point value, i.e. a NaN.
1290	MSK_RES_ERR_NONLINEAR_EQUALITY The model contains a nonlinear equality which defines a nonconvex set.
1055	MSK_RES_ERR_DATA_FILE_EXT The data file format cannot be determined from the file name.
1210	MSK_RES_ERR_PARAM_INDEX Parameter index is out of range.
1285	MSK_RES_ERR_FIRSTI Invalid <code>firsti</code> .
1000	MSK_RES_ERR_LICENSE Invalid license.
1299	MSK_RES_ERR_ARGUMENT_PERM_ARRAY An invalid permutation array is specified.
85	MSK_RES_WRN_LP_DROP_VARIABLE Ignored a variable because the variable was not previously defined. Usually this implies that a variable appears in the bound section but not in the objective or the constraints.
1287	MSK_RES_ERR_FIRSTJ Invalid <code>firstj</code> .
1432	MSK_RES_ERR_USER_NLO_FUNC The user-defined nonlinear function reported an error.
1219	MSK_RES_ERR_INF_DOU_INDEX A double information index is out of range for the specified type.
1286	MSK_RES_ERR_LASTI Invalid <code>lasti</code> .
1431	MSK_RES_ERR_USER_FUNC_RET_DATA An user function returned invalid data.
3900	MSK_RES_ERR_SIZE_LICENSE_NUMCORES The computer contains more cpu cores than the license allows for.
1199	MSK_RES_ERR_NR_ARGUMENTS Incorrect number of function arguments.
1293	MSK_RES_ERR_CON_Q_NOT_PSD The quadratic constraint matrix is not positive semi-definite as expected for a constraint with finite upper bound. This results in a nonconvex problem.
63	MSK_RES_WRN_ZERO_AIJ

continued on next page

continued from previous page	
	One or more zero elements are specified in A.
2504	MSK_RES_ERR_INV_NUMJ Invalid numj.
1650	MSK_RES_ERR_FACTOR An error occurred while factorizing a matrix.
3201	MSK_RES_ERR_INVALID_BRANCH_PRIORITY An invalid branching priority is specified. It should be nonnegative.
1216	MSK_RES_ERR_PARAM_IS_TOO_SMALL The parameter value is too small.
1163	MSK_RES_ERR_LP_WRITE_CONIC_PROBLEM The problem contains cones that cannot be written to an LP formatted file.
1002	MSK_RES_ERR_LICENSE_VERSION The license is valid for another version of MOSEK.
1240	MSK_RES_ERR_MAXNUMCON The maximum number of constraints specified is smaller than the number of constraints in the task.
1050	MSK_RES_ERR_UNKNOWN Unknown error.
1162	MSK_RES_ERR_READ_LP_NONEXISTING_NAME A variable never occurred in objective or constraints.
2503	MSK_RES_ERR_INV_NUMI Invalid numi.
1292	MSK_RES_ERR_NONLINEAR_RANGED The model contains a nonlinear ranged constraint which by definition defines a nonconvex set.
1047	MSK_RES_ERR_THREAD_MUTEX_UNLOCK Could not unlock a mutex.
1100	MSK_RES_ERR_MPS_FILE An error occurred while reading an MPS file.
1156	MSK_RES_ERR_WRITE_OPF_INVALID_VAR_NAME Empty variable names cannot be written to OPF files.
1152	MSK_RES_ERR_LP_DUP_SLACK_NAME The name of the slack variable added to a ranged constraint already exists.
2000	MSK_RES_ERR_NO_PRIMAL_INFEAS_CER A certificate of primal infeasibility is not available.
1158	MSK_RES_ERR_WRITE_LP_FORMAT Problem cannot be written as an LP file.

continued on next page

continued from previous page	
1461	MSK_RES_ERR_NAN_IN_BLC <i>l^c</i> contains an invalid floating point value, i.e. a NaN.
3058	MSK_RES_ERR_SEN_NUMERICAL Numerical difficulties encountered performing the sensitivity analysis.
3052	MSK_RES_ERR_SEN_INDEX_RANGE Index out of range in the sensitivity analysis file.
1027	MSK_RES_ERR_LICENSE_NO_SERVER_SUPPORT The license server does not support the requested feature. Possible reasons for this error include: <ul style="list-style-type: none"> • 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 <code>lmgrd.log</code> .
66	MSK_RES_WRN_SPAR_MAX_LEN A value for a string parameter is longer than the buffer that is supposed to hold it.
3050	MSK_RES_ERR_SEN_FORMAT Syntax error in sensitivity analysis file.
1407	MSK_RES_ERR_INV_QCON_VAL Invalid value in <code>qcval</code> .
1206	MSK_RES_ERR_PARAM_NAME_DOU The parameter name is not correct for a double parameter.
1172	MSK_RES_ERR_OPF_PREMATURE_EOF Premature end of file in an OPF file.
1300	MSK_RES_ERR_CONE_INDEX An index of a non-existing cone has been specified.
1470	MSK_RES_ERR_NAN_IN_C <i>c</i> contains an invalid floating point value, i.e. a NaN.
1066	MSK_RES_ERR_LIVING_TASKS All tasks associated with an enviroment must be deleted before the environment is deleted. There are still some undeleted tasks.
1304	MSK_RES_ERR_MAXNUMCONE

continued on next page

	continued from previous page
	The value specified for <code>maxnumcone</code> is too small.
1103	<code>MSK_RES_ERR_MPS_NULL_CON_NAME</code> An empty constraint name is used in an MPS file.
1417	<code>MSK_RES_ERR_QCON_UPPER_TRIANGLE</code> An element in the upper triangle of a Q^k is specified. Only elements in the lower triangle should be specified.
1171	<code>MSK_RES_ERR_LP_INVALID_CON_NAME</code> A constraint name is invalid when used in an LP formatted file.
1125	<code>MSK_RES_ERR_MPS_TAB_IN_FIELD2</code> A tab char occurred in field 2.
270	<code>MSK_RES_WRN_MIO_INFEASIBLE_FINAL</code> The final mixed-integer problem with all the integer variables fixed at their optimal values is infeasible.
710	<code>MSK_RES_WRN_ZEROS_IN_SPARSE_COL</code> One or more (near) zero elements are specified in a sparse column of a matrix. It is redundant to specify zero elements. Hence, it may indicate an error.
1433	<code>MSK_RES_ERR_USER_NLO_EVAL</code> The user-defined nonlinear function reported an error.
1232	<code>MSK_RES_ERR_INF_TYPE</code> The information type is invalid.
800	<code>MSK_RES_WRN_INCOMPLETE_LINEAR_DEPENDENCY_CHECK</code> The linear dependency check(s) was not completed and therefore the A matrix may contain linear dependencies.
503	<code>MSK_RES_WRN_USING_GENERIC_NAMES</code> The file writer reverts to generic names because a name is blank.
1127	<code>MSK_RES_ERR_MPS_TAB_IN_FIELD5</code> A tab char occurred in field 5.
1056	<code>MSK_RES_ERR_INVALID_FILE_NAME</code> An invalid file name has been specified.
804	<code>MSK_RES_WRN_WRITE_DISCARDED_CFIX</code> The fixed objective term could not be converted to a variable and was discarded in the output file.
1415	<code>MSK_RES_ERR_QOBJ_UPPER_TRIANGLE</code> An element in the upper triangle of Q^o is specified. Only elements in the lower triangle should be specified.
1054	<code>MSK_RES_ERR_FILE_WRITE</code> File write error.
1048	<code>MSK_RES_ERR_THREAD_CREATE</code>

continued on next page

continued from previous page	
	Could not create a thread. This error may occur if a large number of environments are created and not deleted again. In any case it is a good practice to minimize the number of environments created.
1243	MSK_RES_ERR_MAXNUMQNZ The maximum number of non-zeros specified for the Q matrices is smaller than the number of non-zeros in the current Q matrices.
2506	MSK_RES_ERR_CANNOT_HANDLE_NL A function cannot handle a task with nonlinear function call-backs.
1600	MSK_RES_ERR_NO_BASIS_SOL No basic solution is defined.
1131	MSK_RES_ERR_ORD_INVALID Invalid content in branch ordering file.
1303	MSK_RES_ERR_CONE_REP_VAR A variable is included multiple times in the cone.
1075	MSK_RES_ERR_INVALID_OBJ_NAME An invalid objective name is specified.
1052	MSK_RES_ERR_FILE_OPEN Error while opening a file.
250	MSK_RES_WRN_IGNORE_INTEGER Ignored integer constraints.
1296	MSK_RES_ERR_OBJ_Q_NOT_NSD The quadratic coefficient matrix in the objective is not negative semi-definite as expected for a maximization problem.
1064	MSK_RES_ERR_INVALID_TASK The <code>task</code> is invalid.
1065	MSK_RES_ERR_NULL_POINTER An argument to a function is unexpectedly a NULL pointer.
3059	MSK_RES_ERR_CONCURRENT_OPTIMIZER An unsupported optimizer was chosen for use with the concurrent optimizer.
3005	MSK_RES_ERR_API_FATAL_ERROR An internal error occurred in the API. Please report this problem.
1550	MSK_RES_ERR_INV_OPTIMIZER An invalid optimizer has been chosen for the problem. This means that the simplex or the conic optimizer is chosen to optimize a non-linear problem.
1310	MSK_RES_ERR_REMOVE_CONE_VARIABLE A variable cannot be removed because it will make a cone invalid.
62	MSK_RES_WRN_LARGE_AIJ

continued on next page

continued from previous page

	A numerically large value is specified for an $a_{i,j}$ element in A . The parameter MSK_DPAR_DATA_TOL_AIJ_LARGE controls when an $a_{i,j}$ is considered large.
1208	MSK_RES_ERR_PARAM_NAME_STR The parameter name is not correct for a string parameter.
1018	MSK_RES_ERR_LICENSE_FEATURE A requested feature is not available in the license file(s). Most likely due to an incorrect license system setup.
251	MSK_RES_WRN_NO_GLOBAL_OPTIMIZER No global optimizer is available.
1040	MSK_RES_ERR_LINK_FILE_DLL A file cannot be linked to a stream in the DLL version.
1701	MSK_RES_ERR_FEASREPAIR_SOLVING_RELAXED The relaxed problem could not be solved to optimality. Please consult the log file for further details.
1221	MSK_RES_ERR_INDEX_ARR_IS_TOO_SMALL An index in an array argument is too small.
1259	MSK_RES_ERR_SOLVER_PROBTYPE Problem type does not match the chosen optimizer.
1220	MSK_RES_ERR_INF_INT_INDEX An integer information index is out of range for the specified type.
1053	MSK_RES_ERR_FILE_READ File read error.
1440	MSK_RES_ERR_USER_NLO_EVAL_HESSUBI The user-defined nonlinear function reported an invalid subscript in the Hessian.
1441	MSK_RES_ERR_USER_NLO_EVAL_HESSUBJ The user-defined nonlinear function reported an invalid subscript in the Hessian.
300	MSK_RES_WRN_SOL_FILTER Invalid solution filter is specified.
4030	MSK_RES_TRM_INTERNAL The optimizer terminated due to some internal reason. Please contact MOSEK support.
1110	MSK_RES_ERR_MPS_NO_OBJECTIVE No objective is defined in an MPS file.
1403	MSK_RES_ERR_INV_QOBJ_VAL Invalid value in <code>qobjval</code> .
1400	MSK_RES_ERR_INFINITE_BOUND

continued on next page

continued from previous page	
	A numerically huge bound value is specified.
1030	<code>MSK_RES_ERR_OPEN_DL</code> A dynamic link library could not be opened.
3001	<code>MSK_RES_ERR_API_ARRAY_TOO_SMALL</code> An input array was too short.
1046	<code>MSK_RES_ERR_THREAD_MUTEX_LOCK</code> Could not lock a mutex.
1262	<code>MSK_RES_ERR_LAST</code> Invalid index <code>last</code> . A given index was out of expected range.
1151	<code>MSK_RES_ERR_LP_EMPTY</code> The problem cannot be written to an LP formatted file.
1011	<code>MSK_RES_ERR_SIZE_LICENSE_VAR</code> The problem has too many variables to be solved with the available license.
1062	<code>MSK_RES_ERR_INVALID_STREAM</code> An invalid stream is referenced.
2505	<code>MSK_RES_ERR_CANNOT_CLONE_NL</code> A task with a nonlinear function call-back cannot be cloned.
2520	<code>MSK_RES_ERR_INVALID_ACCMODE</code> An invalid access mode is specified.
1250	<code>MSK_RES_ERR_NUMCONLIM</code> Maximum number of constraints limit is exceeded.
2550	<code>MSK_RES_ERR_MBT_INCOMPATIBLE</code> The MBT file is incompatible with this platform. This results from reading a file on a 32 bit platform generated on a 64 bit platform.
1104	<code>MSK_RES_ERR_MPS_NULL_VAR_NAME</code> An empty variable name is used in an MPS file.
72	<code>MSK_RES_WRN_MPS_SPLIT_BOUNDS_VECTOR</code> A <code>BOUNDS</code> vector is split into several nonadjacent parts in an MPS file.
1026	<code>MSK_RES_ERR_LICENSE_SERVER_VERSION</code> The version specified in the checkout request is greater than the highest version number the daemon supports.
1025	<code>MSK_RES_ERR_LICENSE_INVALID_HOSTID</code> The host ID specified in the license file does not match the host ID of the computer.
1045	<code>MSK_RES_ERR_THREAD_MUTEX_INIT</code> Could not initialize a mutex.
54	<code>MSK_RES_WRN_LARGE_CON_FX</code>

continued on next page

continued from previous page	
	An equality constraint is fixed to a numerically large value. This can cause numerical problems.
1280	MSK_RES_ERR_INV_NAME_ITEM An invalid name item code is used.
3106	MSK_RES_ERR_AD_MISSING_RETURN The code list data was invalid. Missing return operation in function.
53	MSK_RES_WRN_LARGE_UP_BOUND A numerically large upper bound value is specified.
3910	MSK_RES_ERR_INFEAS_UNDEFINED The requested value is not defined for this solution type.
901	MSK_RES_WRN_ANA_C_ZERO This warning is issued by the problem analyzer, if the coefficients in the linear part of the objective are all zero.
1112	MSK_RES_ERR_MPS_MUL_CON_NAME A constraint name was specified multiple times in the ROWS section.
1801	MSK_RES_ERR_INVALID_IOMODE Invalid io mode.
1115	MSK_RES_ERR_MPS_INV_SEC_ORDER The sections in the MPS data file are not in the correct order.
1016	MSK_RES_ERR_LICENSE_MAX Maximum number of licenses is reached.
4007	MSK_RES_TRM_USER_CALLBACK The optimizer terminated due to the return of the user-defined call-back function.
805	MSK_RES_WRN_CONSTRUCT_SOLUTION_INFEAS After fixing the integer variables at the suggested values then the problem is infeasible.
1058	MSK_RES_ERR_INVALID_MBT_FILE A MOSEK binary task file is invalid.
1294	MSK_RES_ERR_CON_Q_NOT_NSD The quadratic constraint matrix is not negative semi-definite as expected for a constraint with finite lower bound. This results in a nonconvex problem.
3600	MSK_RES_ERR_XML_INVALID_PROBLEM_TYPE The problem type is not supported by the XML format.
1231	MSK_RES_ERR_INF_INT_NAME An integer information name is invalid.
1107	MSK_RES_ERR_MPS_INV_CON_KEY An invalid constraint key occurred in an MPS file.

continued on next page

continued from previous page	
1425	MSK_RES_ERR_FIXED_BOUND_VALUES A fixed constraint/variable has been specified using the bound keys but the numerical value of the lower and upper bound is different.
4025	MSK_RES_TRM_NUMERICAL_PROBLEM The optimizer terminated due to numerical problems.
3056	MSK_RES_ERR_SEN_INVALID_REGEX Syntax error in regexp or regexp longer than 1024.
52	MSK_RES_WRN_LARGE_LO_BOUND A numerically large lower bound value is specified.
3999	MSK_RES_ERR_API_INTERNAL An internal fatal error occurred in an interface function.
70	MSK_RES_WRN_MPS_SPLIT_RHS_VECTOR An RHS vector is split into several nonadjacent parts in an MPS file.
3053	MSK_RES_ERR_SEN_BOUND_INVALID_UP Analysis of upper bound requested for an index, where no upper bound exists.
1702	MSK_RES_ERR_FEASREPAIR_INCONSISTENT_BOUND The upper bound is less than the lower bound for a variable or a constraint. Please correct this before running the feasibility repair.
1449	MSK_RES_ERR_Y_IS_UNDEFINED The solution item y is undefined.
3200	MSK_RES_ERR_INVALID_BRANCH_DIRECTION An invalid branching direction is specified.
1430	MSK_RES_ERR_USER_FUNC_RET An user function reported an error.
1750	MSK_RES_ERR_NAME_MAX_LEN A name is longer than the buffer that is supposed to hold it.
1305	MSK_RES_ERR_CONE_TYPE Invalid cone type specified.
4005	MSK_RES_TRM_USER_BREAK Not in use.
1256	MSK_RES_ERR_INV_BKC Invalid bound key is specified for a constraint.
4020	MSK_RES_TRM_MAX_NUM_SETBACKS The optimizer terminated as the maximum number of set-backs was reached. This indicates numerical problems and a possibly badly formulated problem.
4015	MSK_RES_TRM_NUM_MAX_NUM_INT_SOLUTIONS

continued on next page

continued from previous page	
	The mixed-integer optimizer terminated as the maximum number of feasible solutions was reached.
3101	MSK_RES_ERR_IDENTICAL_TASKS Some tasks related to this function call were identical. Unique tasks were expected.
1020	MSK_RES_ERR_LICENSE_CANNOT_ALLOCATE The license system cannot allocate the memory required.
904	MSK_RES_WRN_ANA_ALMOST_INT_BOUNDS This warning is issued by the problem analyzer if a constraint is bound nearly integral.
1402	MSK_RES_ERR_INV_QOBJ_SUBJ Invalid value in <code>qosubj</code> .
1302	MSK_RES_ERR_CONE_OVERLAP A new cone which variables overlap with an existing cone has been specified.
807	MSK_RES_WRN_CONSTRUCT_INVALID_SOL_ITG The initial value for one or more of the integer variables is not feasible.
1401	MSK_RES_ERR_INV_QOBJ_SUBI Invalid value in <code>qosubi</code> .
1153	MSK_RES_ERR_WRITE_MPS_INVALID_NAME An invalid name is created while writing an MPS file. Usually this will make the MPS file unreadable.
1553	MSK_RES_ERR_MIO_NOT_LOADED The mixed-integer optimizer is not loaded.
1061	MSK_RES_ERR_NULL_TASK <code>task</code> is a NULL pointer.
1070	MSK_RES_ERR_BLANK_NAME An all blank name has been specified.
1252	MSK_RES_ERR_TOO_SMALL_MAXNUMANZ The maximum number of non-zeros specified for A is smaller than the number of non-zeros in the current A .
1197	MSK_RES_ERR_ARGUMENT_LENNEQ Incorrect length of arguments.
500	MSK_RES_WRN_LICENSE_EXPIRE The license expires.
1200	MSK_RES_ERR_IN_ARGUMENT A function argument is incorrect.
1051	MSK_RES_ERR_SPACE Out of space.

continued on next page

continued from previous page	
1241	MSK_RES_ERR_MAXNUMVAR The maximum number of variables specified is smaller than the number of variables in the task.
1800	MSK_RES_ERR_INVALID_COMPRESSION Invalid compression type.
1101	MSK_RES_ERR_MPS_INV_FIELD A field in the MPS file is invalid. Probably it is too wide.
1060	MSK_RES_ERR_NULL_ENV <code>env</code> is a NULL pointer.
3500	MSK_RES_ERR_INTERNAL_TEST_FAILED An internal unit test function failed.
501	MSK_RES_WRN_LICENSE_SERVER The license server is not responding.
1122	MSK_RES_ERR_MPS_INVALID_OBJSENSE An invalid objective sense is specified.
1168	MSK_RES_ERR_OPF_FORMAT Syntax error in an OPF file
900	MSK_RES_WRN_ANA_LARGE_BOUNDS This warning is issued by the problem analyzer, if one or more constraint or variable bounds are very large. One should consider omitting these bounds entirely by setting them to $+\infty$ or $-\infty$.
1071	MSK_RES_ERR_DUP_NAME The same name was used multiple times for the same problem item type.
1116	MSK_RES_ERR_MPS_MUL_CSEC Multiple CSECTIONs are given the same name.
51	MSK_RES_WRN_LARGE_BOUND A numerically large bound value is specified.
50	MSK_RES_WRN_OPEN_PARAM_FILE The parameter file could not be opened.
1291	MSK_RES_ERR_NONCONVEX The optimization problem is nonconvex.
3100	MSK_RES_ERR_UNB_STEP_SIZE A step size in an optimizer was unexpectedly unbounded. For instance, if the step-size becomes unbounded in phase 1 of the simplex algorithm then an error occurs. Normally this will happen only if the problem is badly formulated. Please contact MOSEK support if this error occurs.
1615	MSK_RES_ERR_BASIS_SINGULAR

continued on next page

continued from previous page	
	The basis is singular and hence cannot be factored.
1155	MSK_RES_ERR_LP_FREE_CONSTRAINT Free constraints cannot be written in LP file format.
1445	MSK_RES_ERR_INVALID_OBJECTIVE_SENSE An invalid objective sense is specified.
0	MSK_RES_OK No error occurred.
3002	MSK_RES_ERR_API_CB_CONNECT Failed to connect a callback object.
1253	MSK_RES_ERR_INV_APTRE <code>aptre[j]</code> is strictly smaller than <code>aptrb[j]</code> for some <code>j</code> .
1013	MSK_RES_ERR_OPTIMIZER_LICENSE The optimizer required is not licensed.
1007	MSK_RES_ERR_FILE_LICENSE Invalid license file.
1160	MSK_RES_ERR_LP_FORMAT Syntax error in an LP file.
1237	MSK_RES_ERR_SOLITEM The solution item number <code>solitem</code> is invalid. Please note that <code>MSK_SOL_ITEM_SNX</code> is invalid for the basic solution.
1010	MSK_RES_ERR_SIZE_LICENSE_CON The problem has too many constraints to be solved with the available license.
1118	MSK_RES_ERR_MPS_CONE_OVERLAP A variable is specified to be a member of several cones.
1090	MSK_RES_ERR_READ_FORMAT The specified format cannot be read.
1408	MSK_RES_ERR_QCON_SUBI_TOO_SMALL Invalid value in <code>qcsubi</code> .
4006	MSK_RES_TRM_STALL

continued on next page

continued from previous page

The optimizer terminated due to slow progress. The most likely reason causing slow progress is that the problem is badly formulated e.g. badly scaled or near infeasible. Sometimes a few dense columns in the constraint matrix can also lead to numerical problems that causes a stall.

The solution returned may or may not be of acceptable quality. Therefore, the solution status should be examined to determine the status of the solution. If the solution is near optimal, then for most practical purposes the solution will be good enough.

In particular, if a linear optimization problem is solved with the interior-point optimizer with basis identification turned on, the returned solution may be of acceptable quality, even in the optimizer stalled.

1580 **MSK_RES_ERR_POSTSOLVE**

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

1215 **MSK_RES_ERR_PARAM_IS_TOO_LARGE**

The parameter value is too large.

1164 **MSK_RES_ERR_LP_WRITE_GECO_PROBLEM**

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

1281 **MSK_RES_ERR_PRO_ITEM**

An invalid problem is used.

1057 **MSK_RES_ERR_INVALID_SOL_FILE_NAME**

An invalid file name has been specified.

1271 **MSK_RES_ERR_INV_CONE_TYPE_STR**

Invalid cone type string encountered.

1283 **MSK_RES_ERR_INVALID_FORMAT_TYPE**

Invalid format type.

57 **MSK_RES_WRN_LARGE_CJ**

A numerically large value is specified for one c_j .

1035 **MSK_RES_ERR_OLDER_DLL**

The dynamic link library is older than the specified version.

1019 **MSK_RES_ERR_PLATFORM_NOT_LICENSED**

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

1119 **MSK_RES_ERR_MPS_CONE_REPEAT**

A variable is repeated within the CSECTION.

3051 **MSK_RES_ERR_SEN_UNDEF_NAME**

An undefined name was encountered in the sensitivity analysis file.

continued on next page

continued from previous page

1380	MSK_RES_ERR_HUGE_AIJ	A numerically huge value is specified for an $a_{i,j}$ element in A . The parameter <code>MSK_DPAR_DATA_TOL_AIJ_HUGE</code> controls when an $a_{i,j}$ is considered huge.
71	MSK_RES_WRN_MPS_SPLIT_RAN_VECTOR	A RANGE vector is split into several nonadjacent parts in an MPS file.
3054	MSK_RES_ERR_SEN_BOUND_INVALID_LO	Analysis of lower bound requested for an index, where no lower bound exists.
3105	MSK_RES_ERR_AD_MISSING_OPERAND	The code list data was invalid. Missing operand for operator.
1111	MSK_RES_ERR_MPS_SPLITTED_VAR	All elements in a column of the A matrix must be specified consecutively. Hence, it is illegal to specify non-zero elements in A for variable 1, then for variable 2 and then variable 1 again.
1080	MSK_RES_ERR_SPACE_LEAKING	MOSEK is leaking memory. This can be due to either an incorrect use of MOSEK or a bug.
1201	MSK_RES_ERR_ARGUMENT_DIMENSION	A function argument is of incorrect dimension.
1159	MSK_RES_ERR_READ_LP_MISSING_END_TAG	Missing End tag in LP file.
4001	MSK_RES_TRM_MAX_TIME	The optimizer terminated at the maximum amount of time.
810	MSK_RES_WRN_CONSTRUCT_NO_SOL_ITG	The construct solution requires an integer solution.
3700	MSK_RES_ERR_INVALID_AMPL_STUB	Invalid AMPL stub.
1260	MSK_RES_ERR_OBJECTIVE_RANGE	Empty objective range.
1238	MSK_RES_ERR_WHICHITEM_NOT_ALLOWED	<code>whichitem</code> is unacceptable.
1471	MSK_RES_ERR_NAN_IN_BLX	l^x contains an invalid floating point value, i.e. a NaN.
1236	MSK_RES_ERR_WHICHSOL	The solution defined by <code>compwhichsol</code> does not exist.
801	MSK_RES_WRN_ELIMINATOR_SPACE	The eliminator is skipped at least once due to lack of space.

continued on next page

continued from previous page	
1049	MSK_RES_ERR_THREAD_COND_INIT Could not initialize a condition.
1269	MSK_RES_ERR_INV_SK_STR Invalid status key string encountered.
1036	MSK_RES_ERR_NEWER_DLL The dynamic link library is newer than the specified version.
1251	MSK_RES_ERR_NUMVARLIM Maximum number of variables limit is exceeded.
1113	MSK_RES_ERR_MPS_MUL_QSEC Multiple QSECTIONs are specified for a constraint in the MPS data file.
502	MSK_RES_WRN_EMPTY_NAME A variable or constraint name is empty. The output file may be invalid.
4003	MSK_RES_TRM_MIO_NEAR_REL_GAP The mixed-integer optimizer terminated because the near optimal relative gap tolerance was satisfied.
80	MSK_RES_WRN_LP_OLD_QUAD_FORMAT Missing $\prime/2\prime$ after quadratic expressions in bound or objective.
1272	MSK_RES_ERR_INV_CONE_TYPE Invalid cone type code is encountered.
1102	MSK_RES_ERR_MPS_INV_MARKER An invalid marker has been specified in the MPS file.
1230	MSK_RES_ERR_INF_DOU_NAME A double information name is invalid.
1264	MSK_RES_ERR_NEGATIVE_APPEND Cannot append a negative number.
1270	MSK_RES_ERR_INV_SK Invalid status key code.
1006	MSK_RES_ERR_PROB_LICENSE The software is not licensed to solve the problem.
3104	MSK_RES_ERR_AD_INVALID_OPERAND The code list data was invalid. An unknown operand was used.
1015	MSK_RES_ERR_LICENSE_SERVER The license server is not responding.
400	MSK_RES_WRN_TOO_FEW_BASIS_VARS An incomplete basis has been specified. Too few basis variables are specified.
1161	MSK_RES_ERR_WRITE_LP_NON_UNIQUE_NAME

continued on next page

continued from previous page	
	An auto-generated name is not unique.
1108	MSK_RES_ERR_MPS_INV_BOUND_KEY An invalid bound key occurred in an MPS file.
1472	MSK_RES_ERR_NAN_IN_BUX u^x contains an invalid floating point value, i.e. a NaN.
1450	MSK_RES_ERR_NAN_IN_DOUBLE_DATA An invalid floating point value was used in some double data.
1109	MSK_RES_ERR_MPS_INV_SEC_NAME An invalid section name occurred in an MPS file.
1266	MSK_RES_ERR_BASIS An invalid basis is specified. Either too many or too few basis variables are specified.
1257	MSK_RES_ERR_INV_BKX An invalid bound key is specified for a variable.
351	MSK_RES_WRN_SOL_FILE_IGNORED_CON One or more lines in the constraint section were ignored when reading a solution file.
902	MSK_RES_WRN_ANA_EMPTY_COLS This warning is issued by the problem analyzer, if columns, in which all coefficients are zero, are found.
1128	MSK_RES_ERR_MPS_INVALID_OBJ_NAME An invalid objective name is specified.
1217	MSK_RES_ERR_PARAM_VALUE_STR The parameter value string is incorrect.
1222	MSK_RES_ERR_INDEX_ARR_IS_TOO_LARGE An index in an array argument is too large.
1306	MSK_RES_ERR_CONE_TYPE_STR Invalid cone type specified.
1405	MSK_RES_ERR_INV_QCON_SUBI Invalid value in qconsubi.
1760	MSK_RES_ERR_NAME_IS_NULL The name buffer is a NULL pointer.
1258	MSK_RES_ERR_INV_VAR_TYPE An invalid variable type is specified for a variable.
1157	MSK_RES_ERR_LP_FILE_FORMAT Syntax error in an LP file.
1021	MSK_RES_ERR_LICENSE_CANNOT_CONNECT MOSEK cannot connect to the license server. Most likely the license server is not up and running.

continued on next page

continued from previous page	
4002	MSK_RES_TRM_OBJECTIVE_RANGE The optimizer terminated on the bound of the objective range.
1126	MSK_RES_ERR_MPS_TAB_IN_FIELD3 A tab char occurred in field 3.
350	MSK_RES_WRN_UNDEF_SOL_FILE_NAME Undefined name occurred in a solution.
1255	MSK_RES_ERR_INV_BK Invalid bound key.
1169	MSK_RES_ERR_OPF_NEW_VARIABLE Introducing new variables is now allowed. When a [variables] section is present, it is not allowed to introduce new variables later in the problem.
1014	MSK_RES_ERR_FLEXLM The FLEXlm license manager reported an error.
1275	MSK_RES_ERR_INVALID_SURPLUS Invalid surplus.
65	MSK_RES_WRN_NAME_MAX_LEN A name is longer than the buffer that is supposed to hold it.
1301	MSK_RES_ERR_CONE_SIZE A cone with too few members is specified.
1261	MSK_RES_ERR_FIRST Invalid first.
1473	MSK_RES_ERR_NAN_IN_AIJ $a_{i,j}$ contains an invalid floating point value, i.e. a NaN.
4031	MSK_RES_TRM_INTERNAL_STOP The optimizer terminated for internal reasons. Please contact MOSEK support.
1117	MSK_RES_ERR_MPS_CONE_TYPE Invalid cone type specified in a CSECTION.
1005	MSK_RES_ERR_SIZE_LICENSE The problem is bigger than the license.
1409	MSK_RES_ERR_QCON_SUBI_TOO_LARGE Invalid value in qcsubi.
1375	MSK_RES_ERR_HUGE_C A huge value in absolute size is specified for one c_j .
1446	MSK_RES_ERR_UNDEFINED_OBJECTIVE_SENSE The objective sense has not been specified before the optimization.
4000	MSK_RES_TRM_MAX_ITERATIONS The optimizer terminated at the maximum number of iterations.

continued on next page

continued from previous page		
802	MSK_RES_WRN_PRESOLVE_OUTOFSPACE	The presolve is incomplete due to lack of space.
1130	MSK_RES_ERR_ORD_INVALID_BRANCH_DIR	An invalid branch direction key is specified.
3103	MSK_RES_ERR_AD_INVALID_OPERATOR	The code list data was invalid. An unknown operator was used.
1166	MSK_RES_ERR_WRITING_FILE	An error occurred while writing file
2502	MSK_RES_ERR_INV_MARKJ	Invalid value in markj.
2500	MSK_RES_ERR_NO_SOLUTION_IN_CALLBACK	The required solution is not available.
2900	MSK_RES_ERR_INVALID_UTF8	An invalid UTF8 string is encountered.
1105	MSK_RES_ERR_MPS_UNDEF_CON_NAME	An undefined constraint name occurred in an MPS file.
1012	MSK_RES_ERR_SIZE_LICENSE_INTVAR	The problem contains too many integer variables to be solved with the available license.
3800	MSK_RES_ERR_INT64_TO_INT32_CAST	An 32 bit integer could not cast to a 64 bit integer.
1552	MSK_RES_ERR_NO_OPTIMIZER_VAR_TYPE	No optimizer is available for this class of optimization problems.

F.41 Response code type

Value	Name	Description
1	MSK_RESPONSE_WRN	The response code is a warning.
2	MSK_RESPONSE_TRM	The response code is an optimizer termination status.
4	MSK_RESPONSE_UNK	The response code does not belong to any class.
0	MSK_RESPONSE_OK	The response code is OK.
3	MSK_RESPONSE_ERR	The response code is an error.

F.42 Scaling type

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

F.43 Scaling type

Value	Name	Description
1	MSK_SCALING_NONE	No scaling is performed.
2	MSK_SCALING_MODERATE	A conservative scaling is performed.
3	MSK_SCALING_AGGRESSIVE	A very aggressive scaling is performed.
0	MSK_SCALING_FREE	The optimizer chooses the scaling heuristic.

F.44 Sensitivity types

Value	Name	Description
1	MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION	Optimal partition sensitivity analysis is performed.
0	MSK_SENSITIVITY_TYPE_BASIS	Basis sensitivity analysis is performed.

F.45 Degeneracy strategies

Value	Name	Description
0	MSK_SIM_DEGEN_NONE	

continued on next page

continued from previous page	
	The simplex optimizer should use no degeneration strategy.
3	MSK_SIM_DEGEN_MODERATE The simplex optimizer should use a moderate degeneration strategy.
4	MSK_SIM_DEGEN_MINIMUM The simplex optimizer should use a minimum degeneration strategy.
2	MSK_SIM_DEGEN_AGGRESSIVE The simplex optimizer should use an aggressive degeneration strategy.
1	MSK_SIM_DEGEN_FREE The simplex optimizer chooses the degeneration strategy.

F.46 Exploit duplicate columns.

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

F.47 Hot-start type employed by the simplex optimizer

Value	Name Description
0	MSK_SIM_HOTSTART_NONE The simplex optimizer performs a coldstart.
2	MSK_SIM_HOTSTART_STATUS_KEYS Only the status keys of the constraints and variables are used to choose the type of hot-start.
1	MSK_SIM_HOTSTART_FREE The simplex optimizer chooses the hot-start type.

F.48 Problem reformulation.

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

F.49 Simplex selection strategy

Value	Name Description
1	MSK_SIM_SELECTION_FULL The optimizer uses full pricing.
5	MSK_SIM_SELECTION_PARTIAL The optimizer uses a partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.
0	MSK_SIM_SELECTION_FREE The optimizer chooses the pricing strategy.
2	MSK_SIM_SELECTION_ASE The optimizer uses approximate steepest-edge pricing.
3	MSK_SIM_SELECTION_DEVEX The optimizer uses devex steepest-edge pricing (or if it is not available an approximate steep-edge selection).
4	MSK_SIM_SELECTION_SE The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

F.50 Solution items

Value	Name Description
4	MSK_SOL_ITEM_SUC

continued on next page

continued from previous page	
	Lagrange multipliers for upper bounds on the constraints.
0	MSK_SOL_ITEM_XC Solution for the constraints.
1	MSK_SOL_ITEM_XX Variable solution.
2	MSK_SOL_ITEM_Y Lagrange multipliers for equations.
5	MSK_SOL_ITEM_SLX Lagrange multipliers for lower bounds on the variables.
6	MSK_SOL_ITEM_SUX Lagrange multipliers for upper bounds on the variables.
7	MSK_SOL_ITEM_SNX Lagrange multipliers corresponding to the conic constraints on the variables.
3	MSK_SOL_ITEM_SLC Lagrange multipliers for lower bounds on the constraints.

F.51 Solution status keys

Value	Name Description
6	MSK_SOL_STA_DUAL_INFEAS_CER The solution is a certificate of dual infeasibility.
5	MSK_SOL_STA_PRIM_INFEAS_CER The solution is a certificate of primal infeasibility.
0	MSK_SOL_STA_UNKNOWN Status of the solution is unknown.
8	MSK_SOL_STA_NEAR_OPTIMAL The solution is nearly optimal.
12	MSK_SOL_STA_NEAR_PRIM_INFEAS_CER The solution is almost a certificate of primal infeasibility.
2	MSK_SOL_STA_PRIM_FEAS The solution is primal feasible.
15	MSK_SOL_STA_NEAR_INTEGER_OPTIMAL The primal solution is near integer optimal.
10	MSK_SOL_STA_NEAR_DUAL_FEAS The solution is nearly dual feasible.
14	MSK_SOL_STA_INTEGER_OPTIMAL

continued on next page

continued from previous page		
		The primal solution is integer optimal.
13	MSK_SOL_STA_NEAR_DUAL_INFEAS_CER	The solution is almost a certificate of dual infeasibility.
11	MSK_SOL_STA_NEAR_PRIM_AND_DUAL_FEAS	The solution is nearly both primal and dual feasible.
1	MSK_SOL_STA_OPTIMAL	The solution is optimal.
4	MSK_SOL_STA_PRIM_AND_DUAL_FEAS	The solution is both primal and dual feasible.
9	MSK_SOL_STA_NEAR_PRIM_FEAS	The solution is nearly primal feasible.
3	MSK_SOL_STA_DUAL_FEAS	The solution is dual feasible.

F.52 Solution types

Value	Name	Description
2	MSK_SOL_ITG	The integer solution.
0	MSK_SOL_ITR	The interior solution.
1	MSK_SOL_BAS	The basic solution.

F.53 Solve primal or dual form

Value	Name	Description
1	MSK_SOLVE_PRIMAL	The optimizer should solve the primal problem.
2	MSK_SOLVE_DUAL	The optimizer should solve the dual problem.
0	MSK_SOLVE_FREE	The optimizer is free to solve either the primal or the dual problem.

F.54 String parameter types

Value	Name	Description
8	MSK_SPAR_PARAM_COMMENT_SIGN	Only the first character in this string is used. It is considered as a start of comment sign in the MOSEK parameter file. Spaces are ignored in the string.
3	MSK_SPAR_FEASREPAIR_NAME_PREFIX	Not applicable.
0	MSK_SPAR_BAS_SOL_FILE_NAME	Name of the <code>bas</code> solution file.
12	MSK_SPAR_READ_MPS_OBJ_NAME	Name of the free constraint used as objective function. An empty name means that the first constraint is used as objective function.
5	MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL	The constraint and variable associated with the total weighted sum of violations are each given the name of this parameter postfixed with <code>CON</code> and <code>VAR</code> respectively.
4	MSK_SPAR_FEASREPAIR_NAME_SEPARATOR	Not applicable.
10	MSK_SPAR_PARAM_WRITE_FILE_NAME	The parameter database is written to this file.
6	MSK_SPAR_INT_SOL_FILE_NAME	Name of the <code>int</code> solution file.
14	MSK_SPAR_READ_MPS_RHS_NAME	Name of the RHS used. An empty name means that the first RHS vector is used.
21	MSK_SPAR_STAT_FILE_NAME	Statistics file name.
24	MSK_SPAR_WRITE_LP_GEN_VAR_NAME	Sometimes when an LP file is written additional variables must be inserted. They will have the prefix denoted by this parameter.
1	MSK_SPAR_DATA_FILE_NAME	Data are read and written to this file.
13	MSK_SPAR_READ_MPS_RAN_NAME	

continued on next page

continued from previous page

	Name of the RANGE vector used. An empty name means that the first RANGE vector is used.
17	MSK_SPAR_SOL_FILTER_XC_LOW A filter used to determine which constraints should be listed in the solution file. A value of “0.5” means that all constraints having $xc[i] > 0.5$ should be listed, whereas “+0.5” means that all constraints having $xc[i] \geq b1c[i] + 0.5$ should be listed. An empty filter means that no filter is applied.
18	MSK_SPAR_SOL_FILTER_XC_UPR 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 $xc[i] \leq buc[i] - 0.5$ should be listed. An empty filter means that no filter is applied.
11	MSK_SPAR_READ_MPS_BOU_NAME Name of the BOUNDS vector used. An empty name means that the first BOUNDS vector is used.
20	MSK_SPAR_SOL_FILTER_XX_UPR A filter used to determine which variables should be listed in the solution file. A value of “0.5” means that all constraints having $xx[j] < 0.5$ should be printed, whereas “-0.5” means all constraints having $xx[j] \leq bux[j] - 0.5$ should be listed. An empty filter means no filter is applied.
23	MSK_SPAR_STAT_NAME Name used when writing the statistics file.
9	MSK_SPAR_PARAM_READ_FILE_NAME Modifications to the parameter database is read from this file.
7	MSK_SPAR_ITR_SOL_FILE_NAME Name of the <code>itr</code> solution file.
15	MSK_SPAR_SENSITIVITY_FILE_NAME Not applicable.
2	MSK_SPAR_DEBUG_FILE_NAME MOSEK debug file.
22	MSK_SPAR_STAT_KEY Key used when writing the summary file.
16	MSK_SPAR_SENSITIVITY_RES_FILE_NAME Not applicable.

continued on next page

continued from previous page

19	MSK_SPAR_SOL_FILTER_XX_LOW
	A filter used to determine which variables should be listed in the solution file. A value of “0.5” means that all constraints having $xx[j] \geq 0.5$ should be listed, whereas “+0.5” means that all constraints having $xx[j] \geq blx[j] + 0.5$ should be listed. An empty filter means no filter is applied.

F.55 Status keys

Value	Name Description
2	MSK_SK_SUPBAS The constraint or variable is super basic.
1	MSK_SK_BAS The constraint or variable is in the basis.
5	MSK_SK_FIX The constraint or variable is fixed.
3	MSK_SK_LOW The constraint or variable is at its lower bound.
6	MSK_SK_INF The constraint or variable is infeasible in the bounds.
0	MSK_SK_UNK The status for the constraint or variable is unknown.
4	MSK_SK_UPR The constraint or variable is at its upper bound.

F.56 Starting point types

Value	Name Description
1	MSK_STARTING_POINT_GUESS The optimizer guesses a starting point.
3	MSK_STARTING_POINT_SATISFY_BOUNDS The starting point is chosen to satisfy all the simple bounds on non-linear variables. If this starting point is employed, then more care than usual should be employed when choosing the bounds on the non-linear variables. In particular very tight bounds should be avoided.

continued on next page

continued from previous page		
2	<code>MSK_STARTING_POINT_CONSTANT</code>	The optimizer constructs a starting point by assigning a constant value to all primal and dual variables. This starting point is normally robust.
0	<code>MSK_STARTING_POINT_FREE</code>	The starting point is chosen automatically.

F.57 Stream types

Value	Name	Description
1	<code>MSK_STREAM_MSG</code>	Message stream. Log information relating to performance and progress of the optimization is written to this stream.
3	<code>MSK_STREAM_WRN</code>	Warning stream. Warning messages are written to this stream.
0	<code>MSK_STREAM_LOG</code>	Log stream. Contains the aggregated contents of all other streams. This means that a message written to any other stream will also be written to this stream.
2	<code>MSK_STREAM_ERR</code>	Error stream. Error messages are written to this stream.

F.58 Integer values

Value	Name	Description
1024	<code>MSK_MAX_STR_LEN</code>	Maximum string length allowed in MOSEK.
20	<code>MSK_LICENSE_BUFFER_LENGTH</code>	The length of a license key buffer.

F.59 Variable types

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

F.60 XML writer output mode

Value	Name	Description
1	MSK_WRITE_XML_MODE_COL	Write in column order.
0	MSK_WRITE_XML_MODE_ROW	Write in row order.

Appendix G

Problem analyzer examples

This appendix presents a few examples of the output produced by the problem analyzer described in Section 13.1. The first two problems are taken from the MIPLIB 2003 collection, <http://miplib.zib.de/>.

G.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, $lb \leq Ax \leq ub$
 distrib: $|b|$ lbs ubs
 [1, 10] 823 823

Variable bounds, $lb \leq x \leq ub$
 distrib: $|b|$ lbs ubs
 0 8904
 [1, 10] 8904

G.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, $lb \leq x \leq ub$

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

G.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

G.4 Problem with both linear and conic constraints

Analyzing the problem

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

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, $lb \leq Ax \leq ub$

distrib:	b	lbs	ubs
	0	21760	21760
	[0.1, 1]		3600

Variable bounds, $lb \leq x \leq ub$

distrib:	b	lbs	ubs
	[1, 10]	3601	3601

Rotated quadratic cones: 3600

dim	RQCs
4	3600

Bibliography

- [1] Richard C. Grinold and Ronald N. Kahn. *Active portfolio management*. McGraw-Hill, New York, 2 edition, 2000.
- [2] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Math. Programming*, 95(1):3–51, 2003.
- [3] E. D. Andersen and K. D. Andersen. Presolving in linear programming. *Math. Programming*, 71(2):221–245, 1995.
- [4] E. D. Andersen and K. D. Andersen. The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In J. B. G. Frenk, C. Roos, T. Terlaky, and S. Zhang, editors, *High Performance Optimization Techniques, Proceedings of the HPOPT-II conference*, 1997. forthcoming.
- [5] E. D. Andersen, J. Gondzio, Cs. Mészáros, and X. Xu. Implementation of interior point methods for large scale linear programming. In T. Terlaky, editor, *Interior-point methods of mathematical programming*, pages 189–252. Kluwer Academic Publishers, 1996.
- [6] E. D. Andersen, C. Roos, and T. Terlaky. On implementing a primal-dual interior-point method for conic quadratic optimization. *Math. Programming*, 95(2), February 2003.
- [7] E. D. Andersen and Y. Ye. Combining interior-point and pivoting algorithms. *Management Sci.*, 42(12):1719–1731, December 1996.
- [8] E. D. Andersen and Y. Ye. A computational study of the homogeneous algorithm for large-scale convex optimization. *Computational Optimization and Applications*, 10:243–269, 1998.
- [9] E. D. Andersen and Y. Ye. On a homogeneous algorithm for the monotone complementarity problem. *Math. Programming*, 84(2):375–399, February 1999.
- [10] Erling D. Andersen. The homogeneous and self-dual model and algorithm for linear optimization. Technical Report TR-1-2009, MOSEK ApS, 2009. <http://www.mosek.com/fileadmin/reports/tech/homolo.pdf>.

- [11] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming: Theory and algorithms*. John Wiley and Sons, New York, 2 edition, 1993.
- [12] C. Beightler and D. T. Phillips. *Applied geometric programming*. John Wiley and Sons, New York, 1976.
- [13] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Math. Programming*, 88(3):411–424, 2000.
- [14] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. MPS/SIAM Series on Optimization. SIAM, 2001.
- [15] S.P. Boyd, S.J. Kim, L. Vandenberghe, and A. Hassibi. A Tutorial on Geometric Programming. Technical report, ISL, Electrical Engineering Department, Stanford University, Stanford, CA, 2004. Available at http://www.stanford.edu/~boyd/gp_tutorial.html.
- [16] V. Chvátal. *Linear programming*. W.H. Freeman and Company, 1983.
- [17] N. Gould and P. L. Toint. Preprocessing for quadratic programming. *Math. Programming*, 100(1):95–132, 2004.
- [18] J. L. Kenningon and K. R. Lewis. Generalized networks: The theory of preprocessing and an empirical analysis. *INFORMS Journal on Computing*, 16(2):162–173, 2004.
- [19] M. S. Lobo, L. Vanderberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra Appl.*, 284:193–228, November 1998.
- [20] M. S. Lobo and M. Fazel, and S. Boyd. Portfolio optimization with linear and fixed transaction costs. Technical report, CDS, California Institute of Technology, 2005. To appear in Annals of Operations Research. <http://www.cds.caltech.edu/~maryam/portfolio.html>.
- [21] J. L. Nazareth. *Computer Solution of Linear Programs*. Oxford University Press, New York, 1987.
- [22] C. Roos, T. Terlaky, and J. -Ph. Vial. *Theory and algorithms for linear optimization: an interior point approach*. John Wiley and Sons, New York, 1997.
- [23] Bernd Scherer. *Portfolio construction and risk budgeting*. Risk Books, 2 edition, 2004.
- [24] S. W. Wallace. Decision making under uncertainty: Is sensitivity of any use. *Oper. Res.*, 48(1):20–25, January 2000.

- [25] H. P. Williams. *Model building in mathematical programming*. John Wiley and Sons, 3 edition, 1993.
- [26] L. A. Wolsey. *Integer programming*. John Wiley and Sons, 1998.

Index

MOSEK / Matlab integration, 21

absolute value, 141

`alloc_add_qnz` (parameter), 305

`ana_sol_basis` (parameter), 305

`ana_sol_infeas_tol` (parameter), 270

`ana_sol_print_violated` (parameter), 306

`auto_sort_a_before_opt` (parameter), 306

`auto_update_sol_info` (parameter), 306

`bas_sol_file_name` (parameter), 378

basis identification, 153

`basis_rel_tol_s` (parameter), 271

`basis_solve_use_plus_one` (parameter), 307

`basis_tol_s` (parameter), 271

`basis_tol_x` (parameter), 271

`bi_clean_optimizer` (parameter), 307

`bi_ignore_max_iter` (parameter), 308

`bi_ignore_num_error` (parameter), 308

`bi_max_iterations` (parameter), 308

bounds, infinite, 122

`cache_license` (parameter), 309

`cache_size_l1` (parameter), 309

`cache_size_l2` (parameter), 309

call-back, 71

iteration, 72

log, 71

`callback_freq` (parameter), 272

certificate

dual, 124

primal, 124

`check_convexity` (parameter), 310

`check_convexity_rel_tol` (parameter), 272

`check_task_data` (parameter), 310

complementarity conditions, 123

concurrent optimization, 161

concurrent solution, 160

`concurrent_num_optimizers` (parameter), 310

`concurrent_priority_dual_simplex` (parameter), 311

`concurrent_priority_free_simplex` (parameter), 311

`concurrent_priority_intpnt` (parameter), 311

`concurrent_priority_primal_simplex` (parameter), 312

conic, 30

optimization, 127

problem, 127

conic modelling, 129

minimizing norms, example, 130

pitfalls, 136

quadratic objective, example, 129

risk and market impact, example

Markowitz model, example, 142

conic optimization, 30

conic quadratic optimization, 30

constraint

matrix, 121, 138, 199

quadratic, 125

constraints

lower limit, 122, 139, 199

upper limit, 122, 139, 199

continuous relaxation, 165

`cpu_type` (parameter), 312

data structures, 75

callback, 83

cones, 80

duasen, 82

info, 83

names, 80

prisen, 81

prob, 75

sol, 81

- symbcon, 83
- data_check (parameter), 312
- data_file_name (parameter), 378
- data_tol_ajj (parameter), 272
- data_tol_ajj_huge (parameter), 273
- data_tol_ajj_large (parameter), 273
- data_tol_bound_inf (parameter), 273
- data_tol_bound_wrn (parameter), 273
- data_tol_c_huge (parameter), 274
- data_tol_cj_large (parameter), 274
- data_tol_qij (parameter), 274
- data_tol_x (parameter), 275
- debug_file_name (parameter), 378
- dual certificate, 124
- dual infeasible, 122, 124
- duality gap (linear problem), 123
- dualizer, 149
- eliminator, 148
- Embedded network flow problems, 158
- feasible, primal, 122
- feasrepair_name_wsumviol (parameter), 379
- feasrepair_optimize (parameter), 313
- feasrepair_tol (parameter), 275
- geometric optimization, 47, 114
- help desk, 19
- hot-start, 155
- infeas_generic_names (parameter), 313
- infeas_prefer_primal (parameter), 313
- infeas_report_auto (parameter), 314
- infeas_report_level (parameter), 314
- infeasible, 175
 - dual, 124
 - primal, 124
- infeasible problems, 175
- infeasible, dual, 122
- infeasible, primal, 122
- infinite bounds, 122
- int_sol_file_name (parameter), 379
- integer optimization, 165
 - relaxation, 165
- interior-point optimizer, 150, 158, 159
- interior-point or simplex optimizer, 157
- intpnt_basis (parameter), 314
- intpnt_co_tol_dfeas (parameter), 275
- intpnt_co_tol_infeas (parameter), 276
- intpnt_co_tol_mu_red (parameter), 276
- intpnt_co_tol_near_rel (parameter), 276
- intpnt_co_tol_pfeas (parameter), 277
- intpnt_co_tol_rel_gap (parameter), 277
- intpnt_diff_step (parameter), 315
- intpnt_factor_debug_lvl (parameter), 315
- intpnt_factor_method (parameter), 316
- intpnt_max_iterations (parameter), 316
- intpnt_max_num_cor (parameter), 316
- intpnt_max_num_refinement_steps (parameter), 316
- intpnt_nl_merit_bal (parameter), 277
- intpnt_nl_tol_dfeas (parameter), 277
- intpnt_nl_tol_mu_red (parameter), 278
- intpnt_nl_tol_near_rel (parameter), 278
- intpnt_nl_tol_pfeas (parameter), 278
- intpnt_nl_tol_rel_gap (parameter), 279
- intpnt_nl_tol_rel_step (parameter), 279
- intpnt_num_threads (parameter), 317
- intpnt_off_col_trh (parameter), 317
- intpnt_order_method (parameter), 317
- intpnt_regularization_use (parameter), 318
- intpnt_scaling (parameter), 318
- intpnt_solve_form (parameter), 319
- intpnt_starting_point (parameter), 319
- intpnt_tol_dfeas (parameter), 279
- intpnt_tol_dsafe (parameter), 279
- intpnt_tol_infeas (parameter), 280
- intpnt_tol_mu_red (parameter), 280
- intpnt_tol_path (parameter), 280
- intpnt_tol_pfeas (parameter), 281
- intpnt_tol_psafe (parameter), 281
- intpnt_tol_rel_gap (parameter), 281
- intpnt_tol_rel_step (parameter), 281
- intpnt_tol_step_size (parameter), 282
- itr_sol_file_name (parameter), 379
- lic_trh_expiry_wrn (parameter), 319
- license system, 21
- license_allow_overuse (parameter), 320
- license_cache_time (parameter), 320
- license_check_time (parameter), 320
- license_debug (parameter), 321

- `license_pause_time` (parameter), 321
- `license_suppress_expire_wrns` (parameter), 321
- `license_wait` (parameter), 322
- linear dependency check, 148
- linear problem, 121
- linearity interval, 187
- `log` (parameter), 322
- `log_bi` (parameter), 322
- `log_bi_freq` (parameter), 323
- `log_check_convexity` (parameter), 323
- `log_concurrent` (parameter), 323
- `log_cut_second_opt` (parameter), 324
- `log_factor` (parameter), 324
- `log_feasrepair` (parameter), 324
- `log_file` (parameter), 325
- `log_head` (parameter), 325
- `log_infeas_ana` (parameter), 325
- `log_intpnt` (parameter), 326
- `log_mio` (parameter), 326
- `log_mio_freq` (parameter), 326
- `log_nonconvex` (parameter), 326
- `log_optimizer` (parameter), 327
- `log_order` (parameter), 327
- `log_param` (parameter), 327
- `log_presolve` (parameter), 328
- `log_response` (parameter), 328
- `log_sensitivity` (parameter), 328
- `log_sensitivity_opt` (parameter), 328
- `log_sim` (parameter), 329
- `log_sim_freq` (parameter), 329
- `log_sim_minor` (parameter), 329
- `log_sim_network_freq` (parameter), 330
- `log_storage` (parameter), 330
- `lower_obj_cut` (parameter), 282
- `lower_obj_cut_finite_trh` (parameter), 282
- LP format, 213
- `lp_write_ignore_incompatible_items` (parameter), 330
- `max_num_warnings` (parameter), 331
- `mio_branch_dir` (parameter), 331
- `mio_branch_priorities_use` (parameter), 331
- `mio_construct_sol` (parameter), 332
- `mio_cont_sol` (parameter), 332
- `mio_cut_level_root` (parameter), 333
- `mio_cut_level_tree` (parameter), 333
- `mio_disable_term_time` (parameter), 283
- `mio_feaspump_level` (parameter), 333
- `mio_heuristic_level` (parameter), 334
- `mio_heuristic_time` (parameter), 283
- `mio_hotstart` (parameter), 334
- `mio_keep_basis` (parameter), 334
- `mio_local_branch_number` (parameter), 335
- `mio_max_num_branches` (parameter), 335
- `mio_max_num_relaxs` (parameter), 335
- `mio_max_num_solutions` (parameter), 336
- `mio_max_time` (parameter), 284
- `mio_max_time_aprx_opt` (parameter), 284
- `mio_mode` (parameter), 336
- `mio_near_tol_abs_gap` (parameter), 284
- `mio_near_tol_rel_gap` (parameter), 285
- `mio_node_optimizer` (parameter), 337
- `mio_node_selection` (parameter), 337
- `mio_optimizer_mode` (parameter), 338
- `mio_presolve_aggregate` (parameter), 338
- `mio_presolve_probing` (parameter), 338
- `mio_presolve_use` (parameter), 339
- `mio_rel_add_cut_limited` (parameter), 285
- `mio_rel_gap_const` (parameter), 285
- `mio_root_optimizer` (parameter), 339
- `mio_strong_branch` (parameter), 340
- `mio_tol_abs_gap` (parameter), 286
- `mio_tol_abs_relax_int` (parameter), 286
- `mio_tol_feas` (parameter), 286
- `mio_tol_rel_gap` (parameter), 287
- `mio_tol_rel_relax_int` (parameter), 287
- `mio_tol_x` (parameter), 287
- mixed-integer optimization, 165
- modelling
 - absolute value, 141
 - in cones, 129
 - market impact term, 143
 - Markowitz portfolio optimization, 143
 - minimizing a sum of norms, 130
 - portfolio optimization, 142
 - transaction costs, 143
- monomial, 114
- MPS format, 199
 - BOUNDS, 206
 - COLUMNS, 203
 - free, 211
 - NAME, 201

- OBJNAME, 202
- OBJSENSE, 202
- QSECTION, 205
- RANGES, 204
- RHS, 203
- ROWS, 202
- mskenopt
 - mskenopt, 46
- mskgpopt
 - mskgpopt, 48
- msklpopt
 - msklpopt, 25
- mskqpopt
 - mskqpopt, 28
- mskscopt
 - mskscopt, 51
- Network flow problems
 - embedded, 158
 - optimizing, 157
- nonconvex_max_iterations (parameter), 340
- nonconvex_tol_feas (parameter), 287
- nonconvex_tol_opt (parameter), 288
- objective
 - quadratic, 125
 - vector, 121
- objective vector, 138
- objective_sense (parameter), 340
- OPF format, 221
- opf_max_terms_per_line (parameter), 341
- opf_write_header (parameter), 341
- opf_write_hints (parameter), 341
- opf_write_parameters (parameter), 341
- opf_write_problem (parameter), 342
- opf_write_sol_bas (parameter), 342
- opf_write_sol_itg (parameter), 342
- opf_write_sol_itr (parameter), 343
- opf_write_solutions (parameter), 343
- optimal solution, 123
- optimization
 - conic, 127
 - integer, 165
 - mixed-integer, 165
- optimization toolbox for MATLAB, 23
- optimizer (parameter), 343
- optimizer_max_time (parameter), 288
- optimizers
 - concurrent, 161
 - conic interior-point, 158
 - convex interior-point, 159
 - linear interior-point, 150
 - parallel, 161
 - simplex, 155
- Optimizing
 - network flow problems, 157
- parallel extensions, 160
- parallel interior-point, 150
- parallel optimizers
 - interior point, 150
- parallel solution, 160
- param_comment_sign (parameter), 379
- param_read_case_name (parameter), 344
- param_read_file_name (parameter), 380
- param_read_ign_error (parameter), 344
- param_write_file_name (parameter), 380
- positive semi-definite, 28
- posynomial, 114
- posynomial optimization, 114
- presolve, 147
 - eliminator, 148
 - linear dependency check, 148
- presolve_elim_fill (parameter), 345
- presolve_eliminator_max_num_tries (parameter), 345
- presolve_eliminator_use (parameter), 345
- presolve_level (parameter), 346
- presolve_lindep_use (parameter), 346
- presolve_lindep_work_lim (parameter), 346
- presolve_tol_aij (parameter), 288
- presolve_tol_lin_dep (parameter), 289
- presolve_tol_s (parameter), 289
- presolve_tol_x (parameter), 289
- presolve_use (parameter), 346
- primal feasible, 122
- primal certificate, 124
- primal infeasible, 122, 124
- primal-dual solution, 122
- qcqo_reformulate_rel_drop_tol (parameter), 289
- qo_separable_reformulation (parameter), 347
- quadratic constraint, 125
- quadratic objective, 125

- quadratic optimization, 125
- read_add_anz (parameter), 347
- read_add_con (parameter), 347
- read_add_cone (parameter), 348
- read_add_qnz (parameter), 348
- read_add_var (parameter), 348
- read_anz (parameter), 348
- read_con (parameter), 349
- read_cone (parameter), 349
- read_data_compressed (parameter), 349
- read_data_format (parameter), 350
- read_keep_free_con (parameter), 350
- read_lp_drop_new_vars_in_bou (parameter), 350
- read_lp_quoted_names (parameter), 351
- read_mps_bou_name (parameter), 380
- read_mps_format (parameter), 351
- read_mps_keep_int (parameter), 352
- read_mps_obj_name (parameter), 380
- read_mps_obj_sense (parameter), 352
- read_mps_quoted_names (parameter), 352
- read_mps_ran_name (parameter), 381
- read_mps_relax (parameter), 352
- read_mps_rhs_name (parameter), 381
- read_mps_width (parameter), 353
- read_q_mode (parameter), 353
- read_qnz (parameter), 353
- read_task_ignore_param (parameter), 354
- read_var (parameter), 354
- relaxation, continuous, 165
- scaling, 149
- sensitivity analysis, 185
 - basis type, 187
 - optimal partition type, 188
- sensitivity_optimizer (parameter), 354
- sensitivity_type (parameter), 355
- separable convex optimization, 50
- shadow price, 187
- sim_basis_factor_use (parameter), 355
- sim_degen (parameter), 356
- sim_dual_crash (parameter), 356
- sim_dual_phaseone_method (parameter), 356
- sim_dual_restrict_selection (parameter), 357
- sim_dual_selection (parameter), 357
- sim_exploit_dupvec (parameter), 358
- sim_hotstart (parameter), 358
- sim_hotstart_lu (parameter), 358
- sim_integer (parameter), 359
- sim_lu_tol_rel_piv (parameter), 290
- sim_max_iterations (parameter), 359
- sim_max_num_setbacks (parameter), 359
- sim_network_detect (parameter), 360
- sim_network_detect_hotstart (parameter), 360
- sim_network_detect_method (parameter), 360
- sim_non_singular (parameter), 361
- sim_primal_crash (parameter), 361
- sim_primal_phaseone_method (parameter), 361
- sim_primal_restrict_selection (parameter), 362
- sim_primal_selection (parameter), 362
- sim_refactor_freq (parameter), 363
- sim_reformulation (parameter), 363
- sim_save_lu (parameter), 363
- sim_scaling (parameter), 364
- sim_scaling_method (parameter), 364
- sim_solve_form (parameter), 364
- sim_stability_priority (parameter), 365
- sim_switch_optimizer (parameter), 365
- simplex optimizer, 155
- simplex_abs_tol_piv (parameter), 290
- sol_filter_keep_basic (parameter), 366
- sol_filter_keep_ranged (parameter), 366
- sol_filter_xc_low (parameter), 381
- sol_filter_xc_upr (parameter), 382
- sol_filter_xx_low (parameter), 382
- sol_filter_xx_upr (parameter), 382
- sol_quoted_names (parameter), 366
- sol_read_name_width (parameter), 367
- sol_read_width (parameter), 367
- solution, optimal, 123
- solution, primal-dual, 122
- solution_callback (parameter), 367
- stat_file_name (parameter), 383
- stat_key (parameter), 383
- stat_name (parameter), 383
- symbolic constants
 - MSK_ACC_CON, 385
 - MSK_ACC_VAR, 385
 - MSK_ADOP_ADD, 385
 - MSK_ADOP_DIV, 386
 - MSK_ADOP_EXP, 385
 - MSK_ADOP_LOG, 386
 - MSK_ADOP_MUL, 385

- MSK_ADOP_POW, 385
- MSK_ADOP_RET, 385
- MSK_ADOP_SUB, 385
- MSK_ADOPTYPE_CONSTANT, 386
- MSK_ADOPTYPE_NONE, 386
- MSK_ADOPTYPE_REFERENCE, 386
- MSK_ADOPTYPE_VARIABLE, 386
- MSK_BI_ALWAYS, 386
- MSK_BI_IF_FEASIBLE, 386
- MSK_BI_NEVER, 386
- MSK_BI_NO_ERROR, 386
- MSK_BI_OTHER, 386
- MSK_BK_FR, 387
- MSK_BK_FX, 387
- MSK_BK_LO, 387
- MSK_BK_RA, 387
- MSK_BK_UP, 387
- MSK_BRANCH_DIR_DOWN, 387
- MSK_BRANCH_DIR_FREE, 387
- MSK_BRANCH_DIR_UP, 387
- MSK_CALLBACK_BEGIN_BI, 395
- MSK_CALLBACK_BEGIN_CONCURRENT, 389
- MSK_CALLBACK_BEGIN_CONIC, 389
- MSK_CALLBACK_BEGIN_DUAL_BI, 389
- MSK_CALLBACK_BEGIN_DUAL_SENSITIVITY, 389
- MSK_CALLBACK_BEGIN_DUAL_SETUP_BI, 391
- MSK_CALLBACK_BEGIN_DUAL_SIMPLEX, 392
- MSK_CALLBACK_BEGIN_DUAL_SIMPLEX_BI, 394
- MSK_CALLBACK_BEGIN_FULL_CONVEXITY_CHECK, 389
- MSK_CALLBACK_BEGIN_INFEAS_ANA, 394
- MSK_CALLBACK_BEGIN_INTPNT, 394
- MSK_CALLBACK_BEGIN_LICENSE_WAIT, 389
- MSK_CALLBACK_BEGIN_MIO, 396
- MSK_CALLBACK_BEGIN_NETWORK_DUAL_SIMPLEX, 388
- MSK_CALLBACK_BEGIN_NETWORK_PRIMAL_SIMPLEX, 396
- MSK_CALLBACK_BEGIN_NETWORK_SIMPLEX, 390
- MSK_CALLBACK_BEGIN_NONCONVEX, 392
- MSK_CALLBACK_BEGIN_OPTIMIZER, 393
- MSK_CALLBACK_BEGIN_PRESOLVE, 395
- MSK_CALLBACK_BEGIN_PRIMAL_BI, 389
- MSK_CALLBACK_BEGIN_PRIMAL_DUAL_SIMPLEX, 394
- MSK_CALLBACK_BEGIN_PRIMAL_DUAL_SIMPLEX_BI, 387
- MSK_CALLBACK_BEGIN_PRIMAL_SENSITIVITY, 394
- MSK_CALLBACK_BEGIN_PRIMAL_SETUP_BI, 392
- MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX, 392
- MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX_BI, 391
- MSK_CALLBACK_BEGIN_QCQO_REFORMULATE, 390
- MSK_CALLBACK_BEGIN_READ, 393
- MSK_CALLBACK_BEGIN_SIMPLEX, 393
- MSK_CALLBACK_BEGIN_SIMPLEX_BI, 394
- MSK_CALLBACK_BEGIN_SIMPLEX_NETWORK_DETECT, 392
- MSK_CALLBACK_BEGIN_WRITE, 393
- MSK_CALLBACK_CONIC, 389
- MSK_CALLBACK_DUAL_SIMPLEX, 392
- MSK_CALLBACK_END_BI, 393
- MSK_CALLBACK_END_CONCURRENT, 388
- MSK_CALLBACK_END_CONIC, 390
- MSK_CALLBACK_END_DUAL_BI, 391
- MSK_CALLBACK_END_DUAL_SENSITIVITY, 390
- MSK_CALLBACK_END_DUAL_SETUP_BI, 393
- MSK_CALLBACK_END_DUAL_SIMPLEX, 394
- MSK_CALLBACK_END_DUAL_SIMPLEX_BI, 390
- MSK_CALLBACK_END_FULL_CONVEXITY_CHECK, 395
- MSK_CALLBACK_END_INFEAS_ANA, 391
- MSK_CALLBACK_END_INTPNT, 387
- MSK_CALLBACK_END_LICENSE_WAIT, 391
- MSK_CALLBACK_END_MIO, 388
- MSK_CALLBACK_END_NETWORK_DUAL_SIMPLEX, 388
- MSK_CALLBACK_END_NETWORK_PRIMAL_SIMPLEX, 388
- MSK_CALLBACK_END_NETWORK_SIMPLEX, 389
- MSK_CALLBACK_END_NONCONVEX, 392
- MSK_CALLBACK_END_OPTIMIZER, 390
- MSK_CALLBACK_END_PRESOLVE, 393
- MSK_CALLBACK_END_PRIMAL_BI, 394
- MSK_CALLBACK_END_PRIMAL_DUAL_SIMPLEX, 395
- MSK_CALLBACK_END_PRIMAL_DUAL_SIMPLEX_BI, 393
- MSK_CALLBACK_END_PRIMAL_SENSITIVITY, 393
- MSK_CALLBACK_END_PRIMAL_SETUP_BI, 393
- MSK_CALLBACK_END_PRIMAL_SIMPLEX, 393
- MSK_CALLBACK_END_PRIMAL_SIMPLEX_BI, 390
- MSK_CALLBACK_END_QCQO_REFORMULATE, 394
- MSK_CALLBACK_END_READ, 391
- MSK_CALLBACK_END_SIMPLEX, 394
- MSK_CALLBACK_END_SIMPLEX_BI, 393

- MSK_CALLBACK_END_SIMPLEX_NETWORK_DETECT, 388
- MSK_CALLBACK_END_WRITE, 394
- MSK_CALLBACK_IM_BI, 393
- MSK_CALLBACK_IM_CONIC, 395
- MSK_CALLBACK_IM_DUAL_BI, 388
- MSK_CALLBACK_IM_DUAL_SENSIVITY, 394
- MSK_CALLBACK_IM_DUAL_SIMPLEX, 389
- MSK_CALLBACK_IM_FULL_CONVEXITY_CHECK, 392
- MSK_CALLBACK_IM_INTPNT, 388
- MSK_CALLBACK_IM_LICENSE_WAIT, 390
- MSK_CALLBACK_IM_LU, 390
- MSK_CALLBACK_IM_MIO, 395
- MSK_CALLBACK_IM_MIO_DUAL_SIMPLEX, 395
- MSK_CALLBACK_IM_MIO_INTPNT, 395
- MSK_CALLBACK_IM_MIO_PRESOLVE, 395
- MSK_CALLBACK_IM_MIO_PRIMAL_SIMPLEX, 389
- MSK_CALLBACK_IM_NETWORK_DUAL_SIMPLEX, 393
- MSK_CALLBACK_IM_NETWORK_PRIMAL_SIMPLEX, 389
- MSK_CALLBACK_IM_NONCONVEX, 393
- MSK_CALLBACK_IM_ORDER, 395
- MSK_CALLBACK_IM_PRESOLVE, 391
- MSK_CALLBACK_IM_PRIMAL_BI, 395
- MSK_CALLBACK_IM_PRIMAL_DUAL_SIMPLEX, 392
- MSK_CALLBACK_IM_PRIMAL_SENSIVITY, 394
- MSK_CALLBACK_IM_PRIMAL_SIMPLEX, 388
- MSK_CALLBACK_IM_QO_REFORMULATE, 389
- MSK_CALLBACK_IM_SIMPLEX, 390
- MSK_CALLBACK_IM_SIMPLEX_BI, 392
- MSK_CALLBACK_INTPNT, 391
- MSK_CALLBACK_NEW_INT_MIO, 388
- MSK_CALLBACK_NONCOVEX, 391
- MSK_CALLBACK_PRIMAL_SIMPLEX, 392
- MSK_CALLBACK_QCONE, 394
- MSK_CALLBACK_READ_ADD_ANZ, 392
- MSK_CALLBACK_READ_ADD_CONES, 391
- MSK_CALLBACK_READ_ADD_CONS, 388
- MSK_CALLBACK_READ_ADD_QNZ, 389
- MSK_CALLBACK_READ_ADD_VARS, 390
- MSK_CALLBACK_READ_OPF, 390
- MSK_CALLBACK_READ_OPF_SECTION, 390
- MSK_CALLBACK_UPDATE_DUAL_BI, 389
- MSK_CALLBACK_UPDATE_DUAL_SIMPLEX, 395
- MSK_CALLBACK_UPDATE_DUAL_SIMPLEX_BI, 390
- MSK_CALLBACK_UPDATE_NETWORK_DUAL_SIMPLEX, 390
- MSK_CALLBACK_UPDATE_NETWORK_PRIMAL_SIMPLEX, 395
- MSK_CALLBACK_UPDATE_NONCONVEX, 391
- MSK_CALLBACK_UPDATE_PRESOLVE, 390
- MSK_CALLBACK_UPDATE_PRIMAL_BI, 391
- MSK_CALLBACK_UPDATE_PRIMAL_DUAL_SIMPLEX, 394
- MSK_CALLBACK_UPDATE_PRIMAL_DUAL_SIMPLEX_BI, 391
- MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX, 392
- MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX_BI, 388
- MSK_CALLBACK_WRITE_OPF, 395
- MSK_CHECK_CONVEXITY_FULL, 396
- MSK_CHECK_CONVEXITY_NONE, 396
- MSK_CHECK_CONVEXITY_SIMPLE, 396
- MSK_COMPRESS_FREE, 396
- MSK_COMPRESS_GZIP, 396
- MSK_COMPRESS_NONE, 396
- MSK_CPU_AMD_ATHLON, 397
- MSK_CPU_AMD_OPTERON, 397
- MSK_CPU_GENERIC, 397
- MSK_CPU_HP_PARISC20, 397
- MSK_CPU_INTEL_CORE2, 397
- MSK_CPU_INTEL_ITANIUM2, 397
- MSK_CPU_INTEL_P3, 397
- MSK_CPU_INTEL_P4, 397
- MSK_CPU_INTEL_PM, 397
- MSK_CPU_POWERPC_G5, 397
- MSK_CPU_UNKNOWN, 397
- MSK_CT_QUAD, 396
- MSK_CT_RQUAD, 397
- MSK_DATA_FORMAT_EXTENSION, 398
- MSK_DATA_FORMAT_FREE_MPS, 398
- MSK_DATA_FORMAT_LP, 398
- MSK_DATA_FORMAT_MBT, 398
- MSK_DATA_FORMAT_MPS, 398
- MSK_DATA_FORMAT_OP, 398
- MSK_DATA_FORMAT_XML, 397
- MSK_DINF_BI_CLEAN_DUAL_TIME, 402
- MSK_DINF_BI_CLEAN_PRIMAL_DUAL_TIME, 402
- MSK_DINF_BI_CLEAN_PRIMAL_TIME, 403
- MSK_DINF_BI_CLEAN_TIME, 400
- MSK_DINF_BI_DUAL_TIME, 398
- MSK_DINF_BI_PRIMAL_TIME, 402
- MSK_DINF_BI_TIME, 402
- MSK_DINF_CONCURRENT_TIME, 401

- MSK_DINF_INTPNT_DUAL_FEAS, 401
- MSK_DINF_INTPNT_DUAL_OBJ, 400
- MSK_DINF_INTPNT_FACTOR_NUM_FLOPS, 398
- MSK_DINF_INTPNT_KAP_DIV_TAU, 401
- MSK_DINF_INTPNT_ORDER_TIME, 402
- MSK_DINF_INTPNT_PRIMAL_FEAS, 398
- MSK_DINF_INTPNT_PRIMAL_OBJ, 402
- MSK_DINF_INTPNT_TIME, 400
- MSK_DINF_MIO_CONSTRUCT_SOLUTION_OBJ, 400
- MSK_DINF_MIO_HEURISTIC_TIME, 399
- MSK_DINF_MIO_OBJ_ABS_GAP, 402
- MSK_DINF_MIO_OBJ_BOUND, 401
- MSK_DINF_MIO_OBJ_INT, 401
- MSK_DINF_MIO_OBJ_REL_GAP, 399
- MSK_DINF_MIO_OPTIMIZER_TIME, 398
- MSK_DINF_MIO_ROOT_OPTIMIZER_TIME, 403
- MSK_DINF_MIO_ROOT_PRESOLVE_TIME, 400
- MSK_DINF_MIO_TIME, 398
- MSK_DINF_MIO_USER_OBJ_CUT, 401
- MSK_DINF_OPTIMIZER_TIME, 399
- MSK_DINF_PRESOLVE_ELI_TIME, 398
- MSK_DINF_PRESOLVE_LINDEP_TIME, 399
- MSK_DINF_PRESOLVE_TIME, 399
- MSK_DINF_QCQO_REFORMULATE_TIME, 400
- MSK_DINF_RD_TIME, 398
- MSK_DINF_SIM_DUAL_TIME, 399
- MSK_DINF_SIM_FEAS, 400
- MSK_DINF_SIM_NETWORK_DUAL_TIME, 401
- MSK_DINF_SIM_NETWORK_PRIMAL_TIME, 400
- MSK_DINF_SIM_NETWORK_TIME, 399
- MSK_DINF_SIM_OBJ, 399
- MSK_DINF_SIM_PRIMAL_DUAL_TIME, 403
- MSK_DINF_SIM_PRIMAL_TIME, 400
- MSK_DINF_SIM_TIME, 400
- MSK_DINF_SOL_BAS_DUAL_OBJ, 402
- MSK_DINF_SOL_BAS_MAX_DBI, 401
- MSK_DINF_SOL_BAS_MAX_DEQI, 402
- MSK_DINF_SOL_BAS_MAX_PBI, 399
- MSK_DINF_SOL_BAS_MAX_PEQI, 401
- MSK_DINF_SOL_BAS_MAX_PINTI, 400
- MSK_DINF_SOL_BAS_PRIMAL_OBJ, 399
- MSK_DINF_SOL_INT_MAX_PBI, 401
- MSK_DINF_SOL_INT_MAX_PEQI, 401
- MSK_DINF_SOL_INT_MAX_PINTI, 400
- MSK_DINF_SOL_INT_PRIMAL_OBJ, 402
- MSK_DINF_SOL_ITR_DUAL_OBJ, 400
- MSK_DINF_SOL_ITR_MAX_DBI, 403
- MSK_DINF_SOL_ITR_MAX_DCNI, 399
- MSK_DINF_SOL_ITR_MAX_DEQI, 400
- MSK_DINF_SOL_ITR_MAX_PBI, 399
- MSK_DINF_SOL_ITR_MAX_PCNI, 398
- MSK_DINF_SOL_ITR_MAX_PEQI, 402
- MSK_DINF_SOL_ITR_MAX_PINTI, 399
- MSK_DINF_SOL_ITR_PRIMAL_OBJ, 401
- MSK_DPAR_ANA_SOL_INFEAS_TOL, 405
- MSK_DPAR_BASIS_REL_TOL_S, 407
- MSK_DPAR_BASIS_TOL_S, 403
- MSK_DPAR_BASIS_TOL_X, 408
- MSK_DPAR_CALLBACK_FREQ, 408
- MSK_DPAR_CHECK_CONVEXITY_REL_TOL, 404
- MSK_DPAR_DATA_TOL_AIJ, 405
- MSK_DPAR_DATA_TOL_AIJ_HUGE, 408
- MSK_DPAR_DATA_TOL_AIJ_LARGE, 403
- MSK_DPAR_DATA_TOL_BOUND_INF, 408
- MSK_DPAR_DATA_TOL_BOUND_WRN, 408
- MSK_DPAR_DATA_TOL_C_HUGE, 406
- MSK_DPAR_DATA_TOL_CJ_LARGE, 407
- MSK_DPAR_DATA_TOL_QIJ, 409
- MSK_DPAR_DATA_TOL_X, 405
- MSK_DPAR_FEASREPAIR_TOL, 405
- MSK_DPAR_INTPNT_CO_TOL_DFEAS, 403
- MSK_DPAR_INTPNT_CO_TOL_INFEAS, 407
- MSK_DPAR_INTPNT_CO_TOL_MU_RED, 406
- MSK_DPAR_INTPNT_CO_TOL_NEAR_REL, 408
- MSK_DPAR_INTPNT_CO_TOL_PFEAS, 409
- MSK_DPAR_INTPNT_CO_TOL_REL_GAP, 406
- MSK_DPAR_INTPNT_NL_MERIT_BAL, 408
- MSK_DPAR_INTPNT_NL_TOL_DFEAS, 409
- MSK_DPAR_INTPNT_NL_TOL_MU_RED, 404
- MSK_DPAR_INTPNT_NL_TOL_NEAR_REL, 405
- MSK_DPAR_INTPNT_NL_TOL_PFEAS, 408
- MSK_DPAR_INTPNT_NL_TOL_REL_GAP, 409
- MSK_DPAR_INTPNT_NL_TOL_REL_STEP, 407
- MSK_DPAR_INTPNT_TOL_DFEAS, 409
- MSK_DPAR_INTPNT_TOL_DSAFE, 405
- MSK_DPAR_INTPNT_TOL_INFEAS, 405
- MSK_DPAR_INTPNT_TOL_MU_RED, 406
- MSK_DPAR_INTPNT_TOL_PATH, 407
- MSK_DPAR_INTPNT_TOL_PFEAS, 407
- MSK_DPAR_INTPNT_TOL_PSAFE, 408
- MSK_DPAR_INTPNT_TOL_REL_GAP, 408
- MSK_DPAR_INTPNT_TOL_REL_STEP, 406

- MSK_DPAR_INTPNT_TOL_STEP_SIZE, 407
- MSK_DPAR_LOWER_OBJ_CUT, 406
- MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH, 403
- MSK_DPAR_MIO_DISABLE_TERM_TIME, 406
- MSK_DPAR_MIO_HEURISTIC_TIME, 404
- MSK_DPAR_MIO_MAX_TIME, 403
- MSK_DPAR_MIO_MAX_TIME_APRX_OPT, 407
- MSK_DPAR_MIO_NEAR_TOL_ABS_GAP, 409
- MSK_DPAR_MIO_NEAR_TOL_REL_GAP, 405
- MSK_DPAR_MIO_REL_ADD_CUT_LIMITED, 405
- MSK_DPAR_MIO_REL_GAP_CONST, 407
- MSK_DPAR_MIO_TOL_ABS_GAP, 404
- MSK_DPAR_MIO_TOL_ABS_RELAX_INT, 404
- MSK_DPAR_MIO_TOL_FEAS, 405
- MSK_DPAR_MIO_TOL_REL_GAP, 409
- MSK_DPAR_MIO_TOL_REL_RELAX_INT, 409
- MSK_DPAR_MIO_TOL_X, 406
- MSK_DPAR_NONCONVEX_TOL_FEAS, 404
- MSK_DPAR_NONCONVEX_TOL_OPT, 404
- MSK_DPAR_OPTIMIZER_MAX_TIME, 405
- MSK_DPAR_PRESOLVE_TOL_AIJ, 407
- MSK_DPAR_PRESOLVE_TOL_LIN_DEP, 406
- MSK_DPAR_PRESOLVE_TOL_S, 403
- MSK_DPAR_PRESOLVE_TOL_X, 404
- MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL, 409
- MSK_DPAR_SIM_LU_TOL_REL_PIV, 407
- MSK_DPAR_SIMPLEX_ABS_TOL_PIV, 404
- MSK_DPAR_UPPER_OBJ_CUT, 403
- MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH, 404
- MSK_FEASREPAIR_OPTIMIZE_COMBINED, 409
- MSK_FEASREPAIR_OPTIMIZE_NONE, 409
- MSK_FEASREPAIR_OPTIMIZE_PENALTY, 409
- MSK_FEATURE_PTOM, 410
- MSK_FEATURE_PTON, 410
- MSK_FEATURE_PTOX, 410
- MSK_FEATURE_PTS, 410
- MSK_IINF_ANA_PRO_NUM_CON, 411
- MSK_IINF_ANA_PRO_NUM_CON_EQ, 414
- MSK_IINF_ANA_PRO_NUM_CON_FR, 412
- MSK_IINF_ANA_PRO_NUM_CON_LO, 413
- MSK_IINF_ANA_PRO_NUM_CON_RA, 416
- MSK_IINF_ANA_PRO_NUM_CON_UP, 411
- MSK_IINF_ANA_PRO_NUM_VAR, 413
- MSK_IINF_ANA_PRO_NUM_VAR_BIN, 415
- MSK_IINF_ANA_PRO_NUM_VAR_CONT, 411
- MSK_IINF_ANA_PRO_NUM_VAR_EQ, 415
- MSK_IINF_ANA_PRO_NUM_VAR_FR, 415
- MSK_IINF_ANA_PRO_NUM_VAR_INT, 411
- MSK_IINF_ANA_PRO_NUM_VAR_LO, 413
- MSK_IINF_ANA_PRO_NUM_VAR_RA, 414
- MSK_IINF_ANA_PRO_NUM_VAR_UP, 415
- MSK_IINF_CACHE_SIZE_L1, 413
- MSK_IINF_CACHE_SIZE_L2, 413
- MSK_IINF_CONCURRENT_FASTEST_OPTIMIZER, 415
- MSK_IINF_CPU_TYPE, 414
- MSK_IINF_INTPNT_FACTOR_NUM_OFFCOL, 411
- MSK_IINF_INTPNT_ITER, 411
- MSK_IINF_INTPNT_NUM_THREADS, 415
- MSK_IINF_INTPNT_SOLVE_DUAL, 412
- MSK_IINF_MIO_CONSTRUCT_SOLUTION, 413
- MSK_IINF_MIO_INITIAL_SOLUTION, 415
- MSK_IINF_MIO_NUM_ACTIVE_NODES, 415
- MSK_IINF_MIO_NUM_BRANCH, 414
- MSK_IINF_MIO_NUM_CUTS, 412
- MSK_IINF_MIO_NUM_INT_SOLUTIONS, 414
- MSK_IINF_MIO_NUM_RELAX, 414
- MSK_IINF_MIO_NUMCON, 410
- MSK_IINF_MIO_NUMINT, 412
- MSK_IINF_MIO_NUMVAR, 412
- MSK_IINF_MIO_TOTAL_NUM_BASIS_CUTS, 411
- MSK_IINF_MIO_TOTAL_NUM_BRANCH, 414
- MSK_IINF_MIO_TOTAL_NUM_CARDGUB_CUTS, 412
- MSK_IINF_MIO_TOTAL_NUM CLIQUE_CUTS, 411
- MSK_IINF_MIO_TOTAL_NUM_COEF_REDC_CUTS, 416
- MSK_IINF_MIO_TOTAL_NUM_CONTRA_CUTS, 412
- MSK_IINF_MIO_TOTAL_NUM_CUTS, 416
- MSK_IINF_MIO_TOTAL_NUM_DISAGG_CUTS, 416
- MSK_IINF_MIO_TOTAL_NUM_FLOW_COVER_CUTS, 414
- MSK_IINF_MIO_TOTAL_NUM_GCD_CUTS, 414
- MSK_IINF_MIO_TOTAL_NUM_GOMORY_CUTS, 416
- MSK_IINF_MIO_TOTAL_NUM_GUB_COVER_CUTS, 414
- MSK_IINF_MIO_TOTAL_NUM_KNAPSUR_COVER_CUTS, 411
- MSK_IINF_MIO_TOTAL_NUM_LATTICE_CUTS, 411
- MSK_IINF_MIO_TOTAL_NUM_LIFT_CUTS, 413
- MSK_IINF_MIO_TOTAL_NUM_OBJ_CUTS, 410
- MSK_IINF_MIO_TOTAL_NUM_PLAN_LOC_CUTS, 412
- MSK_IINF_MIO_TOTAL_NUM_RELAX, 415
- MSK_IINF_MIO_USER_OBJ_CUT, 415
- MSK_IINF_OPT_NUMCON, 416
- MSK_IINF_OPT_NUMVAR, 410
- MSK_IINF_OPTIMIZE_RESPONSE, 412

- MSK_IINF_RD_NUMCON, 415
- MSK_IINF_RD_NUMCONE, 415
- MSK_IINF_RD_NUMINTVAR, 410
- MSK_IINF_RD_NUMQ, 411
- MSK_IINF_RD_NUMVAR, 413
- MSK_IINF_RD_PROTYPE, 412
- MSK_IINF_SIM_DUAL_DEG_ITER, 411
- MSK_IINF_SIM_DUAL_HOTSTART, 415
- MSK_IINF_SIM_DUAL_HOTSTART_LU, 415
- MSK_IINF_SIM_DUAL_INF_ITER, 412
- MSK_IINF_SIM_DUAL_ITER, 411
- MSK_IINF_SIM_NETWORK_DUAL_DEG_ITER, 413
- MSK_IINF_SIM_NETWORK_DUAL_HOTSTART, 412
- MSK_IINF_SIM_NETWORK_DUAL_HOTSTART_LU, 414
- MSK_IINF_SIM_NETWORK_DUAL_INF_ITER, 411
- MSK_IINF_SIM_NETWORK_DUAL_ITER, 415
- MSK_IINF_SIM_NETWORK_PRIMAL_DEG_ITER, 411
- MSK_IINF_SIM_NETWORK_PRIMAL_HOTSTART, 416
- MSK_IINF_SIM_NETWORK_PRIMAL_HOTSTART_LU, 415
- MSK_IINF_SIM_NETWORK_PRIMAL_INF_ITER, 414
- MSK_IINF_SIM_NETWORK_PRIMAL_ITER, 413
- MSK_IINF_SIM_NUMCON, 411
- MSK_IINF_SIM_NUMVAR, 410
- MSK_IINF_SIM_PRIMAL_DEG_ITER, 416
- MSK_IINF_SIM_PRIMAL_DUAL_DEG_ITER, 413
- MSK_IINF_SIM_PRIMAL_DUAL_HOTSTART, 412
- MSK_IINF_SIM_PRIMAL_DUAL_HOTSTART_LU, 412
- MSK_IINF_SIM_PRIMAL_DUAL_INF_ITER, 416
- MSK_IINF_SIM_PRIMAL_DUAL_ITER, 411
- MSK_IINF_SIM_PRIMAL_HOTSTART, 414
- MSK_IINF_SIM_PRIMAL_HOTSTART_LU, 414
- MSK_IINF_SIM_PRIMAL_INF_ITER, 414
- MSK_IINF_SIM_PRIMAL_ITER, 415
- MSK_IINF_SIM_SOLVE_DUAL, 410
- MSK_IINF_SOL_BAS_PROSTA, 413
- MSK_IINF_SOL_BAS_SOLSTA, 410
- MSK_IINF_SOL_INT_PROSTA, 413
- MSK_IINF_SOL_INT_SOLSTA, 413
- MSK_IINF_SOL_ITR_PROSTA, 412
- MSK_IINF_SOL_ITR_SOLSTA, 412
- MSK_IINF_STO_NUM_A_CACHE_FLUSHES, 413
- MSK_IINF_STO_NUM_A_REALLOC, 414
- MSK_IINF_STO_NUM_A_TRANSPOSES, 410
- MSK_INF_DOU_TYPE, 416
- MSK_INF_INT_TYPE, 416
- MSK_INF_LINT_TYPE, 416
- MSK_IOMODE_READ, 417
- MSK_IOMODE_READWRITE, 417
- MSK_IOMODE_WRITE, 417
- MSK_IPAR_ALLOC_ADD_QNZ, 431
- MSK_IPAR_ANA_SOL_BASIS, 424
- MSK_IPAR_ANA_SOL_PRINT_VIOLATED, 423
- MSK_IPAR_AUTO_SORT_A_BEFORE_OPT, 431
- MSK_IPAR_AUTO_UPDATE_SOL_INFO, 421
- MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE, 422
- MSK_IPAR_BI_CLEAN_OPTIMIZER, 428
- MSK_IPAR_BI_IGNORE_MAX_ITER, 428
- MSK_IPAR_BI_IGNORE_NUM_ERROR, 422
- MSK_IPAR_BI_MAX_ITERATIONS, 426
- MSK_IPAR_CACHE_LICENSE, 427
- MSK_IPAR_CACHE_SIZE_L1, 433
- MSK_IPAR_CACHE_SIZE_L2, 433
- MSK_IPAR_CHECK_CONVEXITY, 424
- MSK_IPAR_CHECK_TASK_DATA, 419
- MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS, 424
- MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX, 420
- MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX, 430
- MSK_IPAR_CONCURRENT_PRIORITY_INTPNT, 430
- MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX, 427
- MSK_IPAR_CPU_TYPE, 423
- MSK_IPAR_DATA_CHECK, 430
- MSK_IPAR_FEASREPAIR_OPTIMIZE, 421
- MSK_IPAR_INFEAS_GENERIC_NAMES, 433
- MSK_IPAR_INFEAS_PREFER_PRIMAL, 429
- MSK_IPAR_INFEAS_REPORT_AUTO, 417
- MSK_IPAR_INFEAS_REPORT_LEVEL, 433
- MSK_IPAR_INTPNT_BASIS, 429
- MSK_IPAR_INTPNT_DIFF_STEP, 427
- MSK_IPAR_INTPNT_FACTOR_DEBUG_LVL, 425
- MSK_IPAR_INTPNT_FACTOR_METHOD, 435
- MSK_IPAR_INTPNT_MAX_ITERATIONS, 423
- MSK_IPAR_INTPNT_MAX_NUM_COR, 423
- MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS, 423
- MSK_IPAR_INTPNT_NUM_THREADS, 431
- MSK_IPAR_INTPNT_OFF_COL_TRH, 420
- MSK_IPAR_INTPNT_ORDER_METHOD, 418
- MSK_IPAR_INTPNT_REGULARIZATION_USE, 428

- MSK_IPAR_INTPNT_SCALING, 435
- MSK_IPAR_INTPNT_SOLVE_FORM, 428
- MSK_IPAR_INTPNT_STARTING_POINT, 427
- MSK_IPAR_LIC_TRH_EXPIRY_WRN, 421
- MSK_IPAR_LICENSE_ALLOW_OVERUSE, 428
- MSK_IPAR_LICENSE_CACHE_TIME, 426
- MSK_IPAR_LICENSE_CHECK_TIME, 429
- MSK_IPAR_LICENSE_DEBUG, 424
- MSK_IPAR_LICENSE_PAUSE_TIME, 425
- MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS, 432
- MSK_IPAR_LICENSE_WAIT, 424
- MSK_IPAR_LOG, 427
- MSK_IPAR_LOG_BI, 423
- MSK_IPAR_LOG_BI_FREQ, 420
- MSK_IPAR_LOG_CHECK_CONVEXITY, 434
- MSK_IPAR_LOG_CONCURRENT, 423
- MSK_IPAR_LOG_CUT_SECOND_OPT, 429
- MSK_IPAR_LOG_FACTOR, 421
- MSK_IPAR_LOG_FEASREPAIR, 428
- MSK_IPAR_LOG_FILE, 430
- MSK_IPAR_LOG_HEAD, 434
- MSK_IPAR_LOG_INFEAS_ANA, 419
- MSK_IPAR_LOG_INTPNT, 425
- MSK_IPAR_LOG_MIO, 425
- MSK_IPAR_LOG_MIO_FREQ, 419
- MSK_IPAR_LOG_NONCONVEX, 421
- MSK_IPAR_LOG_OPTIMIZER, 431
- MSK_IPAR_LOG_ORDER, 422
- MSK_IPAR_LOG_PARAM, 425
- MSK_IPAR_LOG_PREOLVE, 420
- MSK_IPAR_LOG_RESPONSE, 433
- MSK_IPAR_LOG_SENSITIVITY, 418
- MSK_IPAR_LOG_SENSITIVITY_OPT, 417
- MSK_IPAR_LOG_SIM, 421
- MSK_IPAR_LOG_SIM_FREQ, 421
- MSK_IPAR_LOG_SIM_MINOR, 423
- MSK_IPAR_LOG_SIM_NETWORK_FREQ, 427
- MSK_IPAR_LOG_STORAGE, 429
- MSK_IPAR_LP_WRITE_IGNORE_INCOMPATIBLE_ITEMS, 425
- MSK_IPAR_MAX_NUM_WARNINGS, 432
- MSK_IPAR_MIO_BRANCH_DIR, 425
- MSK_IPAR_MIO_BRANCH_PRIORITIES_USE, 418
- MSK_IPAR_MIO_CONSTRUCT_SOL, 431
- MSK_IPAR_MIO_CONT_SOL, 426
- MSK_IPAR_MIO_CUT_LEVEL_ROOT, 422
- MSK_IPAR_MIO_CUT_LEVEL_TREE, 418
- MSK_IPAR_MIO_FEASPUMP_LEVEL, 432
- MSK_IPAR_MIO_HEURISTIC_LEVEL, 435
- MSK_IPAR_MIO_HOTSTART, 431
- MSK_IPAR_MIO_KEEP_BASIS, 435
- MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER, 420
- MSK_IPAR_MIO_MAX_NUM_BRANCHES, 431
- MSK_IPAR_MIO_MAX_NUM_RELAXS, 435
- MSK_IPAR_MIO_MAX_NUM_SOLUTIONS, 429
- MSK_IPAR_MIO_MODE, 421
- MSK_IPAR_MIO_NODE_OPTIMIZER, 417
- MSK_IPAR_MIO_NODE_SELECTION, 422
- MSK_IPAR_MIO_OPTIMIZER_MODE, 425
- MSK_IPAR_MIO_PREOLVE_AGGREGATE, 425
- MSK_IPAR_MIO_PREOLVE_PROBING, 428
- MSK_IPAR_MIO_PREOLVE_USE, 429
- MSK_IPAR_MIO_ROOT_OPTIMIZER, 420
- MSK_IPAR_MIO_STRONG_BRANCH, 432
- MSK_IPAR_NONCONVEX_MAX_ITERATIONS, 420
- MSK_IPAR_OBJECTIVE_SENSE, 426
- MSK_IPAR_OPF_MAX_TERMS_PER_LINE, 428
- MSK_IPAR_OPF_WRITE_HEADER, 426
- MSK_IPAR_OPF_WRITE_HINTS, 430
- MSK_IPAR_OPF_WRITE_PARAMETERS, 418
- MSK_IPAR_OPF_WRITE_PROBLEM, 434
- MSK_IPAR_OPF_WRITE_SOL_BAS, 419
- MSK_IPAR_OPF_WRITE_SOL_ITG, 418
- MSK_IPAR_OPF_WRITE_SOL_ITR, 418
- MSK_IPAR_OPF_WRITE_SOLUTIONS, 426
- MSK_IPAR_OPTIMIZER, 432
- MSK_IPAR_PARAM_READ_CASE_NAME, 431
- MSK_IPAR_PARAM_READ_IGN_ERROR, 426
- MSK_IPAR_PREOLVE_ELIM_FILL, 420
- MSK_IPAR_PREOLVE_ELIMINATOR_MAX_NUM_TRIES, 424
- MSK_IPAR_PREOLVE_ELIMINATOR_USE, 430
- MSK_IPAR_PREOLVE_LEVEL, 417
- MSK_IPAR_PREOLVE_LINDEP_USE, 426
- MSK_IPAR_PREOLVE_LINDEP_WORK_LIM, 423
- MSK_IPAR_PREOLVE_USE, 417
- MSK_IPAR_QO_SEPARABLE_REFORMULATION, 424
- MSK_IPAR_READ_ADD_ANZ, 418
- MSK_IPAR_READ_ADD_CON, 423
- MSK_IPAR_READ_ADD_CONE, 417
- MSK_IPAR_READ_ADD_QNZ, 420
- MSK_IPAR_READ_ADD_VAR, 417

- MSK_IPAR_READ_ANZ, 421
- MSK_IPAR_READ_CON, 418
- MSK_IPAR_READ_CONE, 432
- MSK_IPAR_READ_DATA_COMPRESSED, 434
- MSK_IPAR_READ_DATA_FORMAT, 434
- MSK_IPAR_READ_KEEP_FREE_CON, 430
- MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOV, 421
- MSK_IPAR_READ_LP_QUOTED_NAMES, 429
- MSK_IPAR_READ_MPS_FORMAT, 431
- MSK_IPAR_READ_MPS_KEEP_INT, 429
- MSK_IPAR_READ_MPS_OBJ_SENSE, 427
- MSK_IPAR_READ_MPS_QUOTED_NAMES, 431
- MSK_IPAR_READ_MPS_RELAX, 418
- MSK_IPAR_READ_MPS_WIDTH, 422
- MSK_IPAR_READ_Q_MODE, 434
- MSK_IPAR_READ_QNZ, 419
- MSK_IPAR_READ_TASK_IGNORE_PARAM, 425
- MSK_IPAR_READ_VAR, 434
- MSK_IPAR_SENSITIVITY_ALL, 428
- MSK_IPAR_SENSITIVITY_OPTIMIZER, 430
- MSK_IPAR_SENSITIVITY_TYPE, 419
- MSK_IPAR_SIM_BASIS_FACTOR_USE, 428
- MSK_IPAR_SIM_DEGEN, 430
- MSK_IPAR_SIM_DUAL_CRASH, 434
- MSK_IPAR_SIM_DUAL_PHASEONE_METHOD, 432
- MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION, 419
- MSK_IPAR_SIM_DUAL_SELECTION, 424
- MSK_IPAR_SIM_EXPLOIT_DUPVEC, 425
- MSK_IPAR_SIM_HOTSTART, 425
- MSK_IPAR_SIM_HOTSTART_LU, 426
- MSK_IPAR_SIM_INTEGER, 432
- MSK_IPAR_SIM_MAX_ITERATIONS, 423
- MSK_IPAR_SIM_MAX_NUM_SETBACKS, 430
- MSK_IPAR_SIM_NETWORK_DETECT, 433
- MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART, 426
- MSK_IPAR_SIM_NETWORK_DETECT_METHOD, 434
- MSK_IPAR_SIM_NON_SINGULAR, 430
- MSK_IPAR_SIM_PRIMAL_CRASH, 431
- MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD, 417
- MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION, 432
- MSK_IPAR_SIM_PRIMAL_SELECTION, 419
- MSK_IPAR_SIM_REFACTOR_FREQ, 424
- MSK_IPAR_SIM_REFORMULATION, 434
- MSK_IPAR_SIM_SAVE_LU, 434
- MSK_IPAR_SIM_SCALING, 427
- MSK_IPAR_SIM_SCALING_METHOD, 421
- MSK_IPAR_SIM_SOLVE_FORM, 424
- MSK_IPAR_SIM_STABILITY_PRIORITY, 417
- MSK_IPAR_SIM_SWITCH_OPTIMIZER, 433
- MSK_IPAR_SOL_FILTER_KEEP_BASIC, 435
- MSK_IPAR_SOL_FILTER_KEEP_RANGED, 428
- MSK_IPAR_SOL_QUOTED_NAMES, 425
- MSK_IPAR_SOL_READ_NAME_WIDTH, 424
- MSK_IPAR_SOL_READ_WIDTH, 423
- MSK_IPAR_SOLUTION_CALLBACK, 421
- MSK_IPAR_TIMING_LEVEL, 422
- MSK_IPAR_WARNING_LEVEL, 420
- MSK_IPAR_WRITE_BAS_CONSTRAINTS, 432
- MSK_IPAR_WRITE_BAS_HEAD, 418
- MSK_IPAR_WRITE_BAS_VARIABLES, 429
- MSK_IPAR_WRITE_DATA_COMPRESSED, 418
- MSK_IPAR_WRITE_DATA_FORMAT, 425
- MSK_IPAR_WRITE_DATA_PARAM, 430
- MSK_IPAR_WRITE_FREE_CON, 420
- MSK_IPAR_WRITE_GENERIC_NAMES, 420
- MSK_IPAR_WRITE_GENERIC_NAMES_IO, 424
- MSK_IPAR_WRITE_INT_CONSTRAINTS, 419
- MSK_IPAR_WRITE_INT_HEAD, 426
- MSK_IPAR_WRITE_INT_VARIABLES, 418
- MSK_IPAR_WRITE_LP_LINE_WIDTH, 424
- MSK_IPAR_WRITE_LP_QUOTED_NAMES, 421
- MSK_IPAR_WRITE_LP_STRICT_FORMAT, 419
- MSK_IPAR_WRITE_LP_TERMS_PER_LINE, 427
- MSK_IPAR_WRITE_MPS_INT, 430
- MSK_IPAR_WRITE_MPS_OBJ_SENSE, 431
- MSK_IPAR_WRITE_MPS_QUOTED_NAMES, 422
- MSK_IPAR_WRITE_MPS_STRICT, 417
- MSK_IPAR_WRITE_PRECISION, 428
- MSK_IPAR_WRITE_SOL_CONSTRAINTS, 420
- MSK_IPAR_WRITE_SOL_HEAD, 432
- MSK_IPAR_WRITE_SOL_VARIABLES, 429
- MSK_IPAR_WRITE_TASK_INC_SOL, 426
- MSK_IPAR_WRITE_XML_MODE, 428
- MSK_LANG_DAN, 435
- MSK_LANG_ENG, 435
- MSK_LICENSE_BUFFER_LENGTH, 474
- MSK_LIINF_BI_CLEAN_DUAL_DEG_ITER, 436
- MSK_LIINF_BI_CLEAN_DUAL_ITER, 436
- MSK_LIINF_BI_CLEAN_PRIMAL_DEG_ITER, 436
- MSK_LIINF_BI_CLEAN_PRIMAL_DUAL_DEG_ITER, 436
- MSK_LIINF_BI_CLEAN_PRIMAL_DUAL_ITER, 436

- MSK_LIINF_BI_CLEAN_PRIMAL_DUAL_SUB_ITER, 436
- MSK_LIINF_BI_CLEAN_PRIMAL_ITER, 436
- MSK_LIINF_BI_DUAL_ITER, 436
- MSK_LIINF_BI_PRIMAL_ITER, 436
- MSK_LIINF_INTPNT_FACTOR_NUM_NZ, 436
- MSK_LIINF_MIO_INTPNT_ITER, 436
- MSK_LIINF_MIO_SIMPLEX_ITER, 436
- MSK_LIINF_RD_NUMANZ, 436
- MSK_LIINF_RD_NUMQNZ, 436
- MSK_MARK_LO, 437
- MSK_MARK_UP, 437
- MSK_MAX_STR_LEN, 474
- MSK_MIO_CONT_SOL_ITG, 437
- MSK_MIO_CONT_SOL_ITG_REL, 437
- MSK_MIO_CONT_SOL_NONE, 437
- MSK_MIO_CONT_SOL_ROOT, 437
- MSK_MIO_MODE_IGNORED, 437
- MSK_MIO_MODE_LAZY, 437
- MSK_MIO_MODE_SATISFIED, 438
- MSK_MIO_NODE_SELECTION_BEST, 438
- MSK_MIO_NODE_SELECTION_FIRST, 438
- MSK_MIO_NODE_SELECTION_FREE, 438
- MSK_MIO_NODE_SELECTION_HYBRID, 438
- MSK_MIO_NODE_SELECTION_PSEUDO, 438
- MSK_MIO_NODE_SELECTION_WORST, 438
- MSK_MPS_FORMAT_FREE, 438
- MSK_MPS_FORMAT_RELAXED, 438
- MSK_MPS_FORMAT_STRICT, 438
- MSK_MSG_MPS_SELECTED, 439
- MSK_MSG_READING_FILE, 439
- MSK_MSG_WRITING_FILE, 439
- MSK_NETWORK_DETECT_ADVANCED, 439
- MSK_NETWORK_DETECT_FREE, 439
- MSK_NETWORK_DETECT_SIMPLE, 439
- MSK_OBJECTIVE_SENSE_MAXIMIZE, 439
- MSK_OBJECTIVE_SENSE_MINIMIZE, 439
- MSK_OBJECTIVE_SENSE_UNDEFINED, 439
- MSK_OFF, 440
- MSK_ON, 440
- MSK_OPTIMIZER_CONCURRENT, 440
- MSK_OPTIMIZER_CONIC, 440
- MSK_OPTIMIZER_DUAL_SIMPLEX, 440
- MSK_OPTIMIZER_FREE, 440
- MSK_OPTIMIZER_FREE_SIMPLEX, 440
- MSK_OPTIMIZER_INTPNT, 440
- MSK_OPTIMIZER_MIXED_INT, 440
- MSK_OPTIMIZER_NONCONVEX, 440
- MSK_OPTIMIZER_PRIMAL_DUAL_SIMPLEX, 440
- MSK_OPTIMIZER_PRIMAL_SIMPLEX, 440
- MSK_OPTIMIZER_QCONE, 440
- MSK_ORDER_METHOD_APPMINLOC1, 441
- MSK_ORDER_METHOD_APPMINLOC2, 441
- MSK_ORDER_METHOD_FREE, 441
- MSK_ORDER_METHOD_GRAPHPAR1, 441
- MSK_ORDER_METHOD_GRAPHPAR2, 441
- MSK_ORDER_METHOD_NONE, 441
- MSK_PAR_DOU_TYPE, 441
- MSK_PAR_INT_TYPE, 441
- MSK_PAR_INVALID_TYPE, 441
- MSK_PAR_STR_TYPE, 441
- MSK_PI_CON, 442
- MSK_PI_CONE, 442
- MSK_PI_VAR, 442
- MSK_PRESOLVE_MODE_FREE, 441
- MSK_PRESOLVE_MODE_OFF, 441
- MSK_PRESOLVE_MODE_ON, 441
- MSK_PRO_STA_DUAL_FEAS, 443
- MSK_PRO_STA_DUAL_INFEAS, 443
- MSK_PRO_STA_ILL_POSED, 443
- MSK_PRO_STA_NEAR_DUAL_FEAS, 443
- MSK_PRO_STA_NEAR_PRIM_AND_DUAL_FEAS, 443
- MSK_PRO_STA_NEAR_PRIM_FEAS, 443
- MSK_PRO_STA_PRIM_AND_DUAL_FEAS, 443
- MSK_PRO_STA_PRIM_AND_DUAL_INFEAS, 443
- MSK_PRO_STA_PRIM_FEAS, 443
- MSK_PRO_STA_PRIM_INFEAS, 443
- MSK_PRO_STA_PRIM_INFEAS_OR_UNBOUNDED, 443
- MSK_PRO_STA_UNKNOWN, 443
- MSK_PROBTYPE_CONIC, 442
- MSK_PROBTYPE_GECO, 442
- MSK_PROBTYPE_LO, 442
- MSK_PROBTYPE_MIXED, 442
- MSK_PROBTYPE_QCQO, 442
- MSK_PROBTYPE_QO, 442
- MSK_Q_READ_ADD, 443
- MSK_Q_READ_DROP_LOWER, 443
- MSK_Q_READ_DROP_UPPER, 444
- MSK_RES_ERR_AD_INVALID_CODELIST, 447
- MSK_RES_ERR_AD_INVALID_OPERAND, 462
- MSK_RES_ERR_AD_INVALID_OPERATOR, 465
- MSK_RES_ERR_AD_MISSING_OPERAND, 461

- MSK_RES_ERR_AD_MISSING_RETURN, 455
- MSK_RES_ERR_API_ARRAY_TOO_SMALL, 454
- MSK_RES_ERR_API_CB_CONNECT, 459
- MSK_RES_ERR_API_FATAL_ERROR, 452
- MSK_RES_ERR_API_INTERNAL, 456
- MSK_RES_ERR_ARGUMENT_DIMENSION, 461
- MSK_RES_ERR_ARGUMENT_LENNEQ, 457
- MSK_RES_ERR_ARGUMENT_PERM_ARRAY, 448
- MSK_RES_ERR_ARGUMENT_TYPE, 447
- MSK_RES_ERR_BASIS, 463
- MSK_RES_ERR_BASIS_FACTOR, 446
- MSK_RES_ERR_BASIS_SINGULAR, 458
- MSK_RES_ERR_BLANK_NAME, 457
- MSK_RES_ERR_CANNOT_CLONE_NL, 454
- MSK_RES_ERR_CANNOT_HANDLE_NL, 452
- MSK_RES_ERR_CON_Q_NOT_NSD, 455
- MSK_RES_ERR_CON_Q_NOT_PSD, 448
- MSK_RES_ERR_CONCURRENT_OPTIMIZER, 452
- MSK_RES_ERR_CONE_INDEX, 450
- MSK_RES_ERR_CONE_OVERLAP, 457
- MSK_RES_ERR_CONE_REP_VAR, 452
- MSK_RES_ERR_CONE_SIZE, 464
- MSK_RES_ERR_CONE_TYPE, 456
- MSK_RES_ERR_CONE_TYPE_STR, 463
- MSK_RES_ERR_DATA_FILE_EXT, 448
- MSK_RES_ERR_DUP_NAME, 458
- MSK_RES_ERR_END_OF_FILE, 447
- MSK_RES_ERR_FACTOR, 449
- MSK_RES_ERR_FEASREPAIR_CANNOT_RELAX, 446
- MSK_RES_ERR_FEASREPAIR_INCONSISTENT_BOUND, 456
- MSK_RES_ERR_FEASREPAIR_SOLVING_RELAXED, 453
- MSK_RES_ERR_FILE_LICENSE, 459
- MSK_RES_ERR_FILE_OPEN, 452
- MSK_RES_ERR_FILE_READ, 453
- MSK_RES_ERR_FILE_WRITE, 451
- MSK_RES_ERR_FIRST, 464
- MSK_RES_ERR_FIRSTI, 448
- MSK_RES_ERR_FIRSTJ, 448
- MSK_RES_ERR_FIXED_BOUND_VALUES, 456
- MSK_RES_ERR_FLEXLM, 464
- MSK_RES_ERR_HUGE_AIJ, 461
- MSK_RES_ERR_HUGE_C, 464
- MSK_RES_ERR_IDENTICAL_TASKS, 457
- MSK_RES_ERR_IN_ARGUMENT, 457
- MSK_RES_ERR_INDEX, 446
- MSK_RES_ERR_INDEX_ARR_IS_TOO_LARGE, 463
- MSK_RES_ERR_INDEX_ARR_IS_TOO_SMALL, 453
- MSK_RES_ERR_INDEX_IS_TOO_LARGE, 446
- MSK_RES_ERR_INDEX_IS_TOO_SMALL, 444
- MSK_RES_ERR_INF_DOU_INDEX, 448
- MSK_RES_ERR_INF_DOU_NAME, 462
- MSK_RES_ERR_INF_INT_INDEX, 453
- MSK_RES_ERR_INF_INT_NAME, 455
- MSK_RES_ERR_INF_LINT_INDEX, 447
- MSK_RES_ERR_INF_LINT_NAME, 445
- MSK_RES_ERR_INF_TYPE, 451
- MSK_RES_ERR_INFEAS_UNDEFINED, 455
- MSK_RES_ERR_INFINITE_BOUND, 453
- MSK_RES_ERR_INT64_TO_INT32_CAST, 465
- MSK_RES_ERR_INTERNAL, 446
- MSK_RES_ERR_INTERNAL_TEST_FAILED, 458
- MSK_RES_ERR_INV_APTRE, 459
- MSK_RES_ERR_INV_BK, 464
- MSK_RES_ERR_INV_BKC, 456
- MSK_RES_ERR_INV_BKX, 463
- MSK_RES_ERR_INV_CONE_TYPE, 462
- MSK_RES_ERR_INV_CONE_TYPE_STR, 460
- MSK_RES_ERR_INV_MARKI, 444
- MSK_RES_ERR_INV_MARKJ, 465
- MSK_RES_ERR_INV_NAME_ITEM, 455
- MSK_RES_ERR_INV_NUMI, 449
- MSK_RES_ERR_INV_NUMJ, 449
- MSK_RES_ERR_INV_OPTIMIZER, 452
- MSK_RES_ERR_INV_PROBLEM, 444
- MSK_RES_ERR_INV_QCON_SUBI, 463
- MSK_RES_ERR_INV_QCON_SUBJ, 447
- MSK_RES_ERR_INV_QCON_SUBK, 447
- MSK_RES_ERR_INV_QCON_VAL, 450
- MSK_RES_ERR_INV_QOBJ_SUBI, 457
- MSK_RES_ERR_INV_QOBJ_SUBJ, 457
- MSK_RES_ERR_INV_QOBJ_VAL, 453
- MSK_RES_ERR_INV_SK, 462
- MSK_RES_ERR_INV_SK_STR, 462
- MSK_RES_ERR_INV_SKC, 446
- MSK_RES_ERR_INV_SKN, 445
- MSK_RES_ERR_INV_SKX, 444
- MSK_RES_ERR_INV_VAR_TYPE, 463
- MSK_RES_ERR_INVALID_ACCMODE, 454
- MSK_RES_ERR_INVALID_AMPL_STUB, 461
- MSK_RES_ERR_INVALID_BRANCH_DIRECTION, 456
- MSK_RES_ERR_INVALID_BRANCH_PRIORITY, 449

- MSK_RES_ERR_INVALID_COMPRESSION, 458
- MSK_RES_ERR_INVALID_FILE_NAME, 451
- MSK_RES_ERR_INVALID_FORMAT_TYPE, 460
- MSK_RES_ERR_INVALID_IOMODE, 455
- MSK_RES_ERR_INVALID_MBT_FILE, 455
- MSK_RES_ERR_INVALID_NAME_IN_SOL_FILE, 445
- MSK_RES_ERR_INVALID_OBJ_NAME, 452
- MSK_RES_ERR_INVALID_OBJECTIVE_SENSE, 459
- MSK_RES_ERR_INVALID_SOL_FILE_NAME, 460
- MSK_RES_ERR_INVALID_STREAM, 454
- MSK_RES_ERR_INVALID_SURPLUS, 464
- MSK_RES_ERR_INVALID_TASK, 452
- MSK_RES_ERR_INVALID_UTF8, 465
- MSK_RES_ERR_INVALID_WCHAR, 447
- MSK_RES_ERR_LAST, 454
- MSK_RES_ERR_LASTI, 448
- MSK_RES_ERR_LASTJ, 445
- MSK_RES_ERR_LICENSE, 448
- MSK_RES_ERR_LICENSE_CANNOT_ALLOCATE, 457
- MSK_RES_ERR_LICENSE_CANNOT_CONNECT, 463
- MSK_RES_ERR_LICENSE_EXPIRED, 445
- MSK_RES_ERR_LICENSE_FEATURE, 453
- MSK_RES_ERR_LICENSE_INVALID_HOSTID, 454
- MSK_RES_ERR_LICENSE_MAX, 455
- MSK_RES_ERR_LICENSE_MOSEKLM_DAEMON, 447
- MSK_RES_ERR_LICENSE_NO_SERVER_SUPPORT, 450
- MSK_RES_ERR_LICENSE_SERVER, 462
- MSK_RES_ERR_LICENSE_SERVER_VERSION, 454
- MSK_RES_ERR_LICENSE_VERSION, 449
- MSK_RES_ERR_LINK_FILE_DLL, 453
- MSK_RES_ERR_LIVING_TASKS, 450
- MSK_RES_ERR_LP_DUP_SLACK_NAME, 449
- MSK_RES_ERR_LP_EMPTY, 454
- MSK_RES_ERR_LP_FILE_FORMAT, 463
- MSK_RES_ERR_LP_FORMAT, 459
- MSK_RES_ERR_LP_FREE_CONSTRAINT, 459
- MSK_RES_ERR_LP_INCOMPATIBLE, 446
- MSK_RES_ERR_LP_INVALID_CON_NAME, 451
- MSK_RES_ERR_LP_INVALID_VAR_NAME, 446
- MSK_RES_ERR_LP_WRITE_CONIC_PROBLEM, 449
- MSK_RES_ERR_LP_WRITE_GECO_PROBLEM, 460
- MSK_RES_ERR_LU_MAX_NUM_TRIES, 446
- MSK_RES_ERR_MAXNUMCON, 449
- MSK_RES_ERR_MAXNUMCONE, 450
- MSK_RES_ERR_MAXNUMQNZ, 452
- MSK_RES_ERR_MAXNUMVAR, 458
- MSK_RES_ERR_MBT_INCOMPATIBLE, 454
- MSK_RES_ERR_MIO_NO_OPTIMIZER, 444
- MSK_RES_ERR_MIO_NOT_LOADED, 457
- MSK_RES_ERR_MISSING_LICENSE_FILE, 445
- MSK_RES_ERR_MIXED_PROBLEM, 446
- MSK_RES_ERR_MPS_CONE_OVERLAP, 459
- MSK_RES_ERR_MPS_CONE_REPEAT, 460
- MSK_RES_ERR_MPS_CONE_TYPE, 464
- MSK_RES_ERR_MPS_FILE, 449
- MSK_RES_ERR_MPS_INV_BOUND_KEY, 463
- MSK_RES_ERR_MPS_INV_CON_KEY, 455
- MSK_RES_ERR_MPS_INV_FIELD, 458
- MSK_RES_ERR_MPS_INV_MARKER, 462
- MSK_RES_ERR_MPS_INV_SEC_NAME, 463
- MSK_RES_ERR_MPS_INV_SEC_ORDER, 455
- MSK_RES_ERR_MPS_INVALID_OBJ_NAME, 463
- MSK_RES_ERR_MPS_INVALID_OBJSENSE, 458
- MSK_RES_ERR_MPS_MUL_CON_NAME, 455
- MSK_RES_ERR_MPS_MUL_CSEC, 458
- MSK_RES_ERR_MPS_MUL_QOBJ, 445
- MSK_RES_ERR_MPS_MUL_QSEC, 462
- MSK_RES_ERR_MPS_NO_OBJECTIVE, 453
- MSK_RES_ERR_MPS_NULL_CON_NAME, 451
- MSK_RES_ERR_MPS_NULL_VAR_NAME, 454
- MSK_RES_ERR_MPS_SPLITTED_VAR, 461
- MSK_RES_ERR_MPS_TAB_IN_FIELD2, 451
- MSK_RES_ERR_MPS_TAB_IN_FIELD3, 464
- MSK_RES_ERR_MPS_TAB_IN_FIELD5, 451
- MSK_RES_ERR_MPS_UNDEF_CON_NAME, 465
- MSK_RES_ERR_MPS_UNDEF_VAR_NAME, 447
- MSK_RES_ERR_MUL_A_ELEMENT, 444
- MSK_RES_ERR_NAME_IS_NULL, 463
- MSK_RES_ERR_NAME_MAX_LEN, 456
- MSK_RES_ERR_NAN_IN_AIJ, 464
- MSK_RES_ERR_NAN_IN_BLC, 450
- MSK_RES_ERR_NAN_IN_BLC, 461
- MSK_RES_ERR_NAN_IN_BUC, 448
- MSK_RES_ERR_NAN_IN_BUC, 463
- MSK_RES_ERR_NAN_IN_C, 450
- MSK_RES_ERR_NAN_IN_DOUBLE_DATA, 463
- MSK_RES_ERR_NEGATIVE_APPEND, 462
- MSK_RES_ERR_NEGATIVE_SURPLUS, 447
- MSK_RES_ERR_NEWER_DLL, 462
- MSK_RES_ERR_NO_BASIS_SOL, 452
- MSK_RES_ERR_NO_DUAL_FOR_ITG_SOL, 446
- MSK_RES_ERR_NO_DUAL_INFEAS_CER, 444

- MSK_RES_ERR_NO_INIT_ENV, 445
- MSK_RES_ERR_NO_OPTIMIZER_VAR_TYPE, 465
- MSK_RES_ERR_NO_PRIMAL_INFEAS_CER, 449
- MSK_RES_ERR_NO_SOLUTION_IN_CALLBACK, 465
- MSK_RES_ERR_NONCONVEX, 458
- MSK_RES_ERR_NONLINEAR_EQUALITY, 448
- MSK_RES_ERR_NONLINEAR_RANGED, 449
- MSK_RES_ERR_NR_ARGUMENTS, 448
- MSK_RES_ERR_NULL_ENV, 458
- MSK_RES_ERR_NULL_POINTER, 452
- MSK_RES_ERR_NULL_TASK, 457
- MSK_RES_ERR_NUMCONLIM, 454
- MSK_RES_ERR_NUMVARLIM, 462
- MSK_RES_ERR_OBJ_Q_NOT_NSD, 452
- MSK_RES_ERR_OBJ_Q_NOT_PSD, 445
- MSK_RES_ERR_OBJECTIVE_RANGE, 461
- MSK_RES_ERR_OLDER_DLL, 460
- MSK_RES_ERR_OPEN_DL, 454
- MSK_RES_ERR_OPF_FORMAT, 458
- MSK_RES_ERR_OPF_NEW_VARIABLE, 464
- MSK_RES_ERR_OPF_PREMATURE_EOF, 450
- MSK_RES_ERR_OPTIMIZER_LICENSE, 459
- MSK_RES_ERR_ORD_INVALID, 452
- MSK_RES_ERR_ORD_INVALID_BRANCH_DIR, 465
- MSK_RES_ERR_OVERFLOW, 446
- MSK_RES_ERR_PARAM_INDEX, 448
- MSK_RES_ERR_PARAM_IS_TOO_LARGE, 460
- MSK_RES_ERR_PARAM_IS_TOO_SMALL, 449
- MSK_RES_ERR_PARAM_NAME, 447
- MSK_RES_ERR_PARAM_NAME_DOU, 450
- MSK_RES_ERR_PARAM_NAME_INT, 446
- MSK_RES_ERR_PARAM_NAME_STR, 453
- MSK_RES_ERR_PARAM_TYPE, 444
- MSK_RES_ERR_PARAM_VALUE_STR, 463
- MSK_RES_ERR_PLATFORM_NOT_LICENSED, 460
- MSK_RES_ERR_POSTSOLVE, 460
- MSK_RES_ERR_PRO_ITEM, 460
- MSK_RES_ERR_PROB_LICENSE, 462
- MSK_RES_ERR_QCON_SUBI_TOO_LARGE, 464
- MSK_RES_ERR_QCON_SUBI_TOO_SMALL, 459
- MSK_RES_ERR_QCON_UPPER_TRIANGLE, 451
- MSK_RES_ERR_QOBJ_UPPER_TRIANGLE, 451
- MSK_RES_ERR_READ_FORMAT, 459
- MSK_RES_ERR_READ_LP_MISSING_END_TAG, 461
- MSK_RES_ERR_READ_LP_NONEXISTING_NAME, 449
- MSK_RES_ERR_REMOVE_CONE_VARIABLE, 452
- MSK_RES_ERR_SEN_BOUND_INVALID_LO, 461
- MSK_RES_ERR_SEN_BOUND_INVALID_UP, 456
- MSK_RES_ERR_SEN_FORMAT, 450
- MSK_RES_ERR_SEN_INDEX_INVALID, 445
- MSK_RES_ERR_SEN_INDEX_RANGE, 450
- MSK_RES_ERR_SEN_INVALID_REGEX, 456
- MSK_RES_ERR_SEN_NUMERICAL, 450
- MSK_RES_ERR_SEN_SOLUTION_STATUS, 446
- MSK_RES_ERR_SEN_UNDEF_NAME, 460
- MSK_RES_ERR_SIZE_LICENSE, 464
- MSK_RES_ERR_SIZE_LICENSE_CON, 459
- MSK_RES_ERR_SIZE_LICENSE_INTVAR, 465
- MSK_RES_ERR_SIZE_LICENSE_NUMCORES, 448
- MSK_RES_ERR_SIZE_LICENSE_VAR, 454
- MSK_RES_ERR_SOL_FILE_INVALID_NUMBER, 446
- MSK_RES_ERR_SOLITEM, 459
- MSK_RES_ERR_SOLVER_PROBTYPE, 453
- MSK_RES_ERR_SPACE, 457
- MSK_RES_ERR_SPACE_LEAKING, 461
- MSK_RES_ERR_SPACE_NO_INFO, 447
- MSK_RES_ERR_THREAD_COND_INIT, 462
- MSK_RES_ERR_THREAD_CREATE, 451
- MSK_RES_ERR_THREAD_MUTEX_INIT, 454
- MSK_RES_ERR_THREAD_MUTEX_LOCK, 454
- MSK_RES_ERR_THREAD_MUTEX_UNLOCK, 449
- MSK_RES_ERR_TOO_SMALL_MAXNUMANZ, 457
- MSK_RES_ERR_UNB_STEP_SIZE, 458
- MSK_RES_ERR_UNDEF_SOLUTION, 445
- MSK_RES_ERR_UNDEFINED_OBJECTIVE_SENSE, 464
- MSK_RES_ERR_UNKNOWN, 449
- MSK_RES_ERR_USER_FUNC_RET, 456
- MSK_RES_ERR_USER_FUNC_RET_DATA, 448
- MSK_RES_ERR_USER_NLO_EVAL, 451
- MSK_RES_ERR_USER_NLO_EVAL_HESSUBI, 453
- MSK_RES_ERR_USER_NLO_EVAL_HESSUBJ, 453
- MSK_RES_ERR_USER_NLO_FUNC, 448
- MSK_RES_ERR_WHICHITEM_NOT_ALLOWED, 461
- MSK_RES_ERR_WHICHSOL, 461
- MSK_RES_ERR_WRITE_LP_FORMAT, 449
- MSK_RES_ERR_WRITE_LP_NON_UNIQUE_NAME, 462
- MSK_RES_ERR_WRITE_MPS_INVALID_NAME, 457
- MSK_RES_ERR_WRITE_OPF_INVALID_VAR_NAME, 449
- MSK_RES_ERR_WRITING_FILE, 465
- MSK_RES_ERR_XML_INVALID_PROBLEM_TYPE, 455
- MSK_RES_ERR_Y_IS_UNDEFINED, 456
- MSK_RES_OK, 459

- MSK_RES_TRM_INTERNAL, 453
- MSK_RES_TRM_INTERNAL_STOP, 464
- MSK_RES_TRM_MAX_ITERATIONS, 464
- MSK_RES_TRM_MAX_NUM_SETBACKS, 456
- MSK_RES_TRM_MAX_TIME, 461
- MSK_RES_TRM_MIO_NEAR_ABS_GAP, 444
- MSK_RES_TRM_MIO_NEAR_REL_GAP, 462
- MSK_RES_TRM_MIO_NUM_BRANCHES, 444
- MSK_RES_TRM_MIO_NUM_RELAXS, 447
- MSK_RES_TRM_NUM_MAX_NUM_INT_SOLUTIONS, 456
- MSK_RES_TRM_NUMERICAL_PROBLEM, 456
- MSK_RES_TRM_OBJECTIVE_RANGE, 464
- MSK_RES_TRM_STALL, 459
- MSK_RES_TRM_USER_BREAK, 456
- MSK_RES_TRM_USER_CALLBACK, 455
- MSK_RES_WRN_ANA_ALMOST_INT_BOUNDS, 457
- MSK_RES_WRN_ANA_C_ZERO, 455
- MSK_RES_WRN_ANA_CLOSE_BOUNDS, 445
- MSK_RES_WRN_ANA_EMPTY_COLS, 463
- MSK_RES_WRN_ANA_LARGE_BOUNDS, 458
- MSK_RES_WRN_CONSTRUCT_INVALID_SOL_ITG, 457
- MSK_RES_WRN_CONSTRUCT_NO_SOL_ITG, 461
- MSK_RES_WRN_CONSTRUCT_SOLUTION_INFEAS, 455
- MSK_RES_WRN_DROPPED_NZ_QOBJ, 446
- MSK_RES_WRN_ELIMINATOR_SPACE, 461
- MSK_RES_WRN_EMPTY_NAME, 462
- MSK_RES_WRN_IGNORE_INTEGER, 452
- MSK_RES_WRN_INCOMPLETE_LINEAR_DEPENDENCY_CHECK, 451
- MSK_RES_WRN_LARGE_AIJ, 452
- MSK_RES_WRN_LARGE_BOUND, 458
- MSK_RES_WRN_LARGE_CJ, 460
- MSK_RES_WRN_LARGE_CON_FX, 454
- MSK_RES_WRN_LARGE_LO_BOUND, 456
- MSK_RES_WRN_LARGE_UP_BOUND, 455
- MSK_RES_WRN_LICENSE_EXPIRE, 457
- MSK_RES_WRN_LICENSE_FEATURE_EXPIRE, 447
- MSK_RES_WRN_LICENSE_SERVER, 458
- MSK_RES_WRN_LP_DROP_VARIABLE, 448
- MSK_RES_WRN_LP_OLD_QUAD_FORMAT, 462
- MSK_RES_WRN_MIO_INFEASIBLE_FINAL, 451
- MSK_RES_WRN_MPS_SPLIT_BOU_VECTOR, 454
- MSK_RES_WRN_MPS_SPLIT_RAN_VECTOR, 461
- MSK_RES_WRN_MPS_SPLIT_RHS_VECTOR, 456
- MSK_RES_WRN_NAME_MAX_LEN, 464
- MSK_RES_WRN_NO_GLOBAL_OPTIMIZER, 453
- MSK_RES_WRN_NZ_IN_UPR_TRI, 447
- MSK_RES_WRN_OPEN_PARAM_FILE, 458
- MSK_RES_WRN_PRESOLVE_BAD_PRECISION, 444
- MSK_RES_WRN_PRESOLVE_OUTOFSPACE, 465
- MSK_RES_WRN_SOL_FILE_IGNORED_CON, 463
- MSK_RES_WRN_SOL_FILE_IGNORED_VAR, 444
- MSK_RES_WRN_SOL_FILTER, 453
- MSK_RES_WRN_SPAR_MAX_LEN, 450
- MSK_RES_WRN_TOO_FEW_BASIS_VARS, 462
- MSK_RES_WRN_TOO_MANY_BASIS_VARS, 447
- MSK_RES_WRN_UNDEF_SOL_FILE_NAME, 464
- MSK_RES_WRN_USING_GENERIC_NAMES, 451
- MSK_RES_WRN_WRITE_DISCARDED_CFIX, 451
- MSK_RES_WRN_ZERO_AIJ, 448
- MSK_RES_WRN_ZEROS_IN_SPARSE_COL, 451
- MSK_RES_WRN_ZEROS_IN_SPARSE_ROW, 447
- MSK_RESPONSE_ERR, 465
- MSK_RESPONSE_OK, 465
- MSK_RESPONSE_TRM, 465
- MSK_RESPONSE_UNK, 465
- MSK_RESPONSE_WRN, 465
- MSK_SCALING_AGGRESSIVE, 466
- MSK_SCALING_FREE, 466
- MSK_SCALING_METHOD_FREE, 466
- MSK_SCALING_METHOD_POW2, 466
- MSK_SCALING_MODERATE, 466
- MSK_SCALING_NONE, 466
- MSK_SENSITIVITY_TYPE_BASIS, 466
- MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION, 466
- MSK_SIM_DEGEN_AGGRESSIVE, 467
- MSK_SIM_DEGEN_FREE, 467
- MSK_SIM_DEGEN_MINIMUM, 467
- MSK_SIM_DEGEN_MODERATE, 467
- MSK_SIM_DEGEN_NONE, 466
- MSK_SIM_EXPLOIT_DUPVEC_FREE, 467
- MSK_SIM_EXPLOIT_DUPVEC_OFF, 467
- MSK_SIM_EXPLOIT_DUPVEC_ON, 467
- MSK_SIM_HOTSTART_FREE, 467
- MSK_SIM_HOTSTART_NONE, 467
- MSK_SIM_HOTSTART_STATUS_KEYS, 467
- MSK_SIM_REFORMULATION_AGGRESSIVE, 468
- MSK_SIM_REFORMULATION_FREE, 468
- MSK_SIM_REFORMULATION_OFF, 468
- MSK_SIM_REFORMULATION_ON, 468
- MSK_SIM_SELECTION_ASE, 468
- MSK_SIM_SELECTION_DEVEX, 468

- MSK_SIM_SELECTION_FREE, 468
- MSK_SIM_SELECTION_FULL, 468
- MSK_SIM_SELECTION_PARTIAL, 468
- MSK_SIM_SELECTION_SE, 468
- MSK_SK_BAS, 473
- MSK_SK_FIX, 473
- MSK_SK_INF, 473
- MSK_SK_LOW, 473
- MSK_SK_SUPBAS, 473
- MSK_SK_UNK, 473
- MSK_SK_UPR, 473
- MSK_SOL_BAS, 470
- MSK_SOL_ITEM_SLC, 469
- MSK_SOL_ITEM_SLX, 469
- MSK_SOL_ITEM_SNX, 469
- MSK_SOL_ITEM_SUC, 468
- MSK_SOL_ITEM_SUX, 469
- MSK_SOL_ITEM_XC, 469
- MSK_SOL_ITEM_XX, 469
- MSK_SOL_ITEM_Y, 469
- MSK_SOL_ITG, 470
- MSK_SOL_ITR, 470
- MSK_SOL_STA_DUAL_FEAS, 470
- MSK_SOL_STA_DUAL_INFEAS_CER, 469
- MSK_SOL_STA_INTEGER_OPTIMAL, 469
- MSK_SOL_STA_NEAR_DUAL_FEAS, 469
- MSK_SOL_STA_NEAR_DUAL_INFEAS_CER, 470
- MSK_SOL_STA_NEAR_INTEGER_OPTIMAL, 469
- MSK_SOL_STA_NEAR_OPTIMAL, 469
- MSK_SOL_STA_NEAR_PRIM_AND_DUAL_FEAS, 470
- MSK_SOL_STA_NEAR_PRIM_FEAS, 470
- MSK_SOL_STA_NEAR_PRIM_INFEAS_CER, 469
- MSK_SOL_STA_OPTIMAL, 470
- MSK_SOL_STA_PRIM_AND_DUAL_FEAS, 470
- MSK_SOL_STA_PRIM_FEAS, 469
- MSK_SOL_STA_PRIM_INFEAS_CER, 469
- MSK_SOL_STA_UNKNOWN, 469
- MSK_SOLVE_DUAL, 470
- MSK_SOLVE_FREE, 470
- MSK_SOLVE_PRIMAL, 470
- MSK_SPAR_BAS_SOL_FILE_NAME, 471
- MSK_SPAR_DATA_FILE_NAME, 471
- MSK_SPAR_DEBUG_FILE_NAME, 472
- MSK_SPAR_FEASREPAIR_NAME_PREFIX, 471
- MSK_SPAR_FEASREPAIR_NAME_SEPARATOR, 471
- MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL, 471
- MSK_SPAR_INT_SOL_FILE_NAME, 471
- MSK_SPAR_ITR_SOL_FILE_NAME, 472
- MSK_SPAR_PARAM_COMMENT_SIGN, 471
- MSK_SPAR_PARAM_READ_FILE_NAME, 472
- MSK_SPAR_PARAM_WRITE_FILE_NAME, 471
- MSK_SPAR_READ_MPS_BOU_NAME, 472
- MSK_SPAR_READ_MPS_OBJ_NAME, 471
- MSK_SPAR_READ_MPS_RAN_NAME, 471
- MSK_SPAR_READ_MPS_RHS_NAME, 471
- MSK_SPAR_SENSITIVITY_FILE_NAME, 472
- MSK_SPAR_SENSITIVITY_RES_FILE_NAME, 472
- MSK_SPAR_SOL_FILTER_XC_LOW, 472
- MSK_SPAR_SOL_FILTER_XC_UPR, 472
- MSK_SPAR_SOL_FILTER_XX_LOW, 473
- MSK_SPAR_SOL_FILTER_XX_UPR, 472
- MSK_SPAR_STAT_FILE_NAME, 471
- MSK_SPAR_STAT_KEY, 472
- MSK_SPAR_STAT_NAME, 472
- MSK_SPAR_WRITE_LP_GEN_VAR_NAME, 471
- MSK_STARTING_POINT_CONSTANT, 474
- MSK_STARTING_POINT_FREE, 474
- MSK_STARTING_POINT_GUESS, 473
- MSK_STARTING_POINT_SATISFY_BOUNDS, 473
- MSK_STREAM_ERR, 474
- MSK_STREAM_LOG, 474
- MSK_STREAM_MSG, 474
- MSK_STREAM_WRN, 474
- MSK_VAR_TYPE_CONT, 475
- MSK_VAR_TYPE_INT, 475
- MSK_WRITE_XML_MODE_COL, 475
- MSK_WRITE_XML_MODE_ROW, 475
- timing_level (parameter), 368
- upper_obj_cut (parameter), 290
- upper_obj_cut_finite_trh (parameter), 291
- variables
 - decision, 121, 138, 199
 - lower limit, 122, 139, 199
 - upper limit, 122, 139, 199
- warning_level (parameter), 368
- write_bas_constraints (parameter), 368
- write_bas_head (parameter), 368
- write_bas_variables (parameter), 369
- write_data_compressed (parameter), 369

`write_data_format` (parameter), 369
`write_data_param` (parameter), 370
`write_free_con` (parameter), 370
`write_generic_names` (parameter), 370
`write_generic_names_io` (parameter), 371
`write_int_constraints` (parameter), 371
`write_int_head` (parameter), 371
`write_int_variables` (parameter), 372
`write_lp_gen_var_name` (parameter), 383
`write_lp_line_width` (parameter), 372
`write_lp_quoted_names` (parameter), 372
`write_lp_strict_format` (parameter), 373
`write_lp_terms_per_line` (parameter), 373
`write_mps_int` (parameter), 373
`write_mps_obj_sense` (parameter), 373
`write_mps_quoted_names` (parameter), 374
`write_mps_strict` (parameter), 374
`write_precision` (parameter), 374
`write_sol_constraints` (parameter), 375
`write_sol_head` (parameter), 375
`write_sol_variables` (parameter), 375
`write_task_inc_sol` (parameter), 376
`write_xml_mode` (parameter), 376

xml format, 233