



MOSEK Command Line Tools
Release 10.0.25

MOSEK ApS

06 October 2022

Contents

1	Introduction	1
1.1	Why the Command Line Tools?	2
2	Contact Information	3
3	License Agreement	4
3.1	MOSEK end-user license agreement	4
3.2	Third party licenses	4
4	Installation	10
4.1	Testing the installation	11
5	The Command Line Tool	12
5.1	Introduction	12
5.2	Files	12
5.3	Example	13
5.4	Solver Parameters	15
5.5	Command Line Arguments	16
5.6	The license system	18
6	The MOSEK-bundled AMPL shell	19
6.1	An example	19
6.2	Retrieving solutions	21
6.3	Optimizer options	22
6.4	Hot-start	23
6.5	Infeasibility report	25
6.6	Sensitivity analysis	25
6.7	Using the command line version of the AMPL interface	26
7	Debugging Tutorials	28
7.1	Understanding optimizer log	28
7.2	Addressing numerical issues	33
7.3	Debugging infeasibility	35
7.4	Python Console	40
8	Problem Formulation and Solutions	42
8.1	Continuous problem formulations	42
8.2	Mixed-integer problem formulations	43
9	Optimizers	53
9.1	Presolve	53
9.2	Linear Optimization	55
9.3	Conic Optimization - Interior-point optimizer	62
9.4	The Optimizer for Mixed-Integer Problems	66
10	Additional features	77
10.1	Problem Analyzer	77
10.2	Automatic Repair of Infeasible Problems	78

10.3 Sensitivity Analysis	82
11 API Reference	89
11.1 Parameters grouped by topic	89
11.2 Parameters (alphabetical list sorted by type)	101
11.3 Response codes	142
11.4 Constants	162
11.5 Supported domains	187
12 Supported File Formats	190
12.1 The LP File Format	191
12.2 The MPS File Format	195
12.3 The OPF Format	207
12.4 The CBF Format	217
12.5 The PTF Format	234
12.6 The Task Format	240
12.7 The JSON Format	241
12.8 The Solution File Format	247
13 List of examples	250
14 Interface changes	251
14.1 Important changes compared to version 9	251
14.2 Changes compared to version 9	251
14.3 Parameters compared to version 9	251
14.4 Constants compared to version 9	252
14.5 Response Codes compared to version 9	253
Bibliography	257
Symbol Index	258
Index	266

Chapter 1

Introduction

The **MOSEK** Optimization Suite 10.0.25 is a powerful software package capable of solving large-scale optimization problems of the following kind:

- linear,
- conic:
 - conic quadratic (also known as second-order cone),
 - involving the exponential cone,
 - involving the power cone,
 - semidefinite,
- convex quadratic and quadratically constrained,
- integer.

In order to obtain an overview of features in the **MOSEK** Optimization Suite consult the [product introduction](#) guide.

The most widespread class of optimization problems is *linear optimization problems*, where all relations are linear. The tremendous success of both applications and theory of linear optimization can be ascribed to the following factors:

- The required data are simple, i.e. just matrices and vectors.
- Convexity is guaranteed since the problem is convex by construction.
- Linear functions are trivially differentiable.
- There exist very efficient algorithms and software for solving linear problems.
- Duality properties for linear optimization are nice and simple.

Even if the linear optimization model is only an approximation to the true problem at hand, the advantages of linear optimization may outweigh the disadvantages. In some cases, however, the problem formulation is inherently nonlinear and a linear approximation is either intractable or inadequate. *Conic optimization* has proved to be a very expressive and powerful way to introduce nonlinearities, while preserving all the nice properties of linear optimization listed above.

The fundamental expression in linear optimization is a linear expression of the form

$$Ax - b \geq 0.$$

In conic optimization this is replaced with a wider class of constraints

$$Ax - b \in \mathcal{K}$$

where \mathcal{K} is a *convex cone*. For example in 3 dimensions \mathcal{K} may correspond to an ice cream cone. The conic optimizer in **MOSEK** supports a number of different types of cones \mathcal{K} , which allows a surprisingly large number of nonlinear relations to be modeled, as described in the **MOSEK** [Modeling Cookbook](#), while preserving the nice algorithmic and theoretical properties of linear optimization.

1.1 Why the Command Line Tools?

The **MOSEK** capabilities can be accessed from the command line without the need to use any programming language. The user can input optimization problems using files in a variety of *formats*, or via the AMPL language shell.

The Command Line Tools provides access to:

- Linear Optimization (LO)
- Conic Quadratic (Second-Order Cone) Optimization (CQO, SOCO)
- Power Cone Optimization
- Conic Exponential Optimization (CEO)
- Convex Quadratic and Quadratically Constrained Optimization (QO, QCQO)
- Semidefinite Optimization (SDO)
- Mixed-Integer Optimization (MIO)

as well as to additional utilities for:

- problem analysis,
- sensitivity analysis,
- infeasibility diagnostics.

Chapter 2

Contact Information

Phone	+45 7174 9373	
Website	mosek.com	
Email		
	sales@mosek.com	Sales, pricing, and licensing
	support@mosek.com	Technical support, questions and bug reports
	info@mosek.com	Everything else.
Mailing Address		
	MOSEK ApS	
	Fruebjergvej 3	
	Symbion Science Park, Box 16	
	2100 Copenhagen O	
	Denmark	

You can get in touch with **MOSEK** using popular social media as well:

Blogger	https://blog.mosek.com/
Google Group	https://groups.google.com/forum/#!forum/mosek
Twitter	https://twitter.com/mosektw
Linkedin	https://www.linkedin.com/company/mosek-aps
Youtube	https://www.youtube.com/channel/UCvIyectEVLp31NXeD5mIbEw

In particular **Twitter** is used for news, updates and release announcements.

Chapter 3

License Agreement

3.1 MOSEK end-user license agreement

Before using the **MOSEK** software, please read the license agreement available in the distribution at <MSKHOME>/mosek/10.0/mosek-eula.pdf or on the **MOSEK** website <https://mosek.com/products/license-agreement>. By using **MOSEK** you agree to the terms of that license agreement.

3.2 Third party licenses

MOSEK uses some third-party open-source libraries. Their license details follow.

zlib

MOSEK uses the *zlib* library obtained from the [zlib website](#). The license agreement for *zlib* is shown in [Listing 3.1](#).

Listing 3.1: *zlib* license.

```
zlib.h -- interface of the 'zlib' general purpose compression library
version 1.2.7, May 2nd, 2012

Copyright (C) 1995-2012 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied
warranty. In no event will the authors be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would be
   appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be
   misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly          Mark Adler
jloup@gzip.org            madler@alumni.caltech.edu
```

fplib

MOSEK uses the floating point formatting library developed by David M. Gay obtained from the [netlib website](#). The license agreement for *fplib* is shown in [Listing 3.2](#).

Listing 3.2: *fplib* license.

```
/*
 *
 * The author of this software is David M. Gay.
 *
 * Copyright (c) 1991, 2000, 2001 by Lucent Technologies.
 *
 * Permission to use, copy, modify, and distribute this software for any
 * purpose without fee is hereby granted, provided that this entire notice
 * is included in all copies of any software which is or includes a copy
 * or modification of this software and in all copies of the supporting
 * documentation for such software.
 *
 * THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED
 * WARRANTY. IN PARTICULAR, NEITHER THE AUTHOR NOR LUCENT MAKES ANY
 * REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY
 * OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.
 *
 *****/
```

{fmt}

MOSEK uses the formatting library *{fmt}* developed by Victor Zverovich obtained from [github/fmt](#) and distributed under the MIT license. The license agreement for *{fmt}* is shown in [Listing 3.3](#).

Listing 3.3: *{fmt}* license.

```
Copyright (c) 2012 - present, Victor Zverovich

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the "Software"),
to deal in the Software without restriction, including without limitation
the rights to use, copy, modify, merge, publish, distribute, sublicense,
and/or sell copies of the Software, and to permit persons to whom the Software
is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR
A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```


Zstandard

MOSEK uses the *Zstandard* library developed by Facebook obtained from [github/zstd](https://github.com/facebook/zstd). The license agreement for *Zstandard* is shown in [Listing 3.4](#).

Listing 3.4: *Zstandard* license.

```
BSD License

For Zstandard software

Copyright (c) 2016-present, Facebook, Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name Facebook nor the names of its contributors may be used to
  endorse or promote products derived from this software without specific
  prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

OpenSSL

MOSEK uses the [LibReSSL](#) library, which is build on *OpenSSL*. *OpenSSL* is included under the *OpenSSL* license, [Listing 3.5](#), and the *LibReSSL* additions are licensed under the *ISC* license, [Listing 3.6](#).

Listing 3.5: *OpenSSL* license

```
=====
Copyright (c) 1998-2011 The OpenSSL Project. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in
```

(continues on next page)

the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgment:
"This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (<http://www.openssl.org/>)"
4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.
5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.
6. Redistributions of any form whatsoever must retain the following acknowledgment:
"This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

=====

This product includes cryptographic software written by Eric Young (ey@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Listing 3.6: ISC license

Copyright (C) 1994-2017 Free Software Foundation, Inc.
Copyright (c) 2014 Jeremie Courreges-Anglas <jca@openbsd.org>
Copyright (c) 2014-2015 Joel Sing <jsing@openbsd.org>
Copyright (c) 2014 Ted Unangst <tedu@openbsd.org>
Copyright (c) 2015-2016 Bob Beck <beck@openbsd.org>
Copyright (c) 2015 Marko Kreen <markokr@gmail.com>
Copyright (c) 2015 Reyk Floeter <reyk@openbsd.org>
Copyright (c) 2016 Tobias Pape <tobias@netshed.de>

Permission to use, copy, modify, and/or distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

(continued from previous page)

```
THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL
WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE
AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL
DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR
PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.
```

mimalloc

MOSEK uses the *mimalloc* memory allocator library from [github/mimalloc](https://github.com/mimalloc). The license agreement for *mimalloc* is shown in [Listing 3.7](#).

Listing 3.7: *mimalloc* license.

```
MIT License

Copyright (c) 2019 Microsoft Corporation, Daan Leijen

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

BLASFEO

MOSEK uses the *BLASFEO* linear algebra library developed by Gianluca Frison, obtained from [github/blasfeo](https://github.com/blasfeo). The license agreement for *BLASFEO* is shown in [Listing 3.8](#).

Listing 3.8: *blasfeo* license.

```
BLASFEO -- BLAS For Embedded Optimization.
Copyright (C) 2019 by Gianluca Frison.
Developed at IMTEK (University of Freiburg) under the supervision of Moritz Diehl.
All rights reserved.

The 2-Clause BSD License

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this
```

(continues on next page)

list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

oneTBB

MOSEK uses the *oneTBB* parallelization library which is part of *oneAPI* developed by Intel, obtained from [github/oneTBB](https://github.com/oneTBB), licensed under the Apache License 2.0. The license agreement for *oneTBB* can be found in <https://github.com/oneapi-src/oneTBB/blob/master/LICENSE.txt> .

Chapter 4

Installation

In this section we discuss how to install and setup the **MOSEK** Command Line Tools.

Important: Before running this **MOSEK** interface please make sure that you:

- Installed **MOSEK** correctly. Some operating systems require extra steps. See the [Installation guide](#) for instructions and common troubleshooting tips.
 - Set up a license. See the [Licensing guide](#) for instructions.
-

Locating files in the MOSEK Optimization Suite

The relevant files of the Command Line Tools are organized as reported in [Table 4.1](#).

Table 4.1: Relevant files for the Command Line Tools.

Relative Path	Description	Label
<MSKHOME>/mosek/10.0/tools/platform/<PLATFORM>/bin	Binaries	<BINDIR>
<MSKHOME>/mosek/10.0/tools/platform/<PLATFORM>/bin/mosek	Mosek executable	
<MSKHOME>/mosek/10.0/tools/examples/data	Examples	<EXDIR>

where

- <MSKHOME> is the folder in which the **MOSEK** Optimization Suite has been installed,
- <PLATFORM> is the actual platform among those supported by **MOSEK**, i.e. win32x86, win64x86, linux64x86 or osx64x86.

Setting up paths

The executable file is ready for use. It may be convenient to add the directory <BINDIR> to the environment variable `PATH`, and then **MOSEK** can simply be used by typing

`mosek`

in the command line.

4.1 Testing the installation

To test that Command Line Tools has been installed correctly go to the examples directory <EXDIR> and run **MOSEK** on any of the input files, for example `lo1.mps`:

```
mosek lo1.mps
```

Is should produce output similar to:

```
MOSEK Version 8.0.0.53 (Build date: 2017-1-12 22:21:45)
Copyright (c) MOSEK ApS, Denmark. WWW: mosek.com
Platform: Linux/64-X86

Open file 'lo1.mps'
Reading started.

[....]

Optimizer started.
Interior-point optimizer started.

[....]

Interior-point solution summary
  Problem status  : PRIMAL_AND_DUAL_FEASIBLE
  Solution status : OPTIMAL
  Primal.  obj: 8.3333333280e+01   nrm: 5e+01   Viol.  con: 1e-08   var: 0e+00
  Dual.    obj: 8.3333333242e+01   nrm: 4e+00   Viol.  con: 2e-10   var: 5e-09

Basic solution summary
  Problem status  : PRIMAL_AND_DUAL_FEASIBLE
  Solution status : OPTIMAL
  Primal.  obj: 8.3333333333e+01   nrm: 5e+01   Viol.  con: 7e-15   var: 0e+00
  Dual.    obj: 8.3333333245e+01   nrm: 4e+00   Viol.  con: 2e-10   var: 5e-09

[....]

Open file 'lo1.sol'
Start writing.
done writing. Time: 0.00

Open file 'lo1.bas'
Start writing.
done writing. Time: 0.00

Return code - 0  [MSK_RES_OK]
```

Chapter 5

The Command Line Tool

5.1 Introduction

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

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

where both [options] and [filename] are optional arguments:

- [options] consists of command line arguments that modify the behavior of **MOSEK**. They are listed in [Sec. 5.5](#). In particular, options can be used to set optimizer parameters.
- [filename] is a file describing the optimization problem. The **MOSEK** command line accepts files in any of the *supported file formats* or in the AMPL `.nl` format.

If no arguments are given, **MOSEK** will display a splash screen and exit.

```
user@host:~$ mosek/8/tools/platform/linux64x86/bin/mosek

MOSEK Version 8.0.0.32(BETA) (Build date: 2016-7-12 10:29:26)
Copyright (c) MOSEK ApS, Denmark. WWW: mosek.com
Platform: Linux/64-X86

*** No input file specified. No optimization is performed.

Return code - 0 [MSK_RES_OK]
```

5.2 Files

The **MOSEK** command line tool communicates with the user via files and prints some execution logs and solution summary to the terminal.

Input files

Optimization problems are read from files. See [Sec. 12](#) for details.

File format conversion

To convert between two file formats supported by **MOSEK** use the option `-x` together with `-out` to specify the target file name. The target file type must support the problem type of the source file, otherwise the conversion will be partial. For instance in case a MPS file must be converted in a more readable OPF format, the following line can be used

```
mosek -x -out lo1.opf lo1.mps
```

With the `-x` option the solver will not actually solve the problem.

Output files

Solutions are written to files:

- `.bas` - basic solution,
- `.sol` - interior point solution,
- `.itg` - integer solution (the only available solution for mixed-integer problems).

For linear problems both the basic and interior point solution may be present. Infeasibility certificates are stored in the same files. See [Sec. 12.8](#) for details.

5.3 Example

To solve a problem stored in file, say `lo1.mps`, write:

```
mosek lo1.mps
```

The solver will

- read `lo1.mps` from disk,
- solve the problem and display the solution log and
- store the relevant solution files if any solution exists; file content explained in [Sec. 12.8](#).

```
MOSEK Version 8.0.0.34(BETA) (Build date: 2016-8-24 00:51:13)
Copyright (c) MOSEK ApS, Denmark. WWW: mosek.com
Platform: Linux/64-X86
```

```
Open file '/home/andrea/mosek/8/tools/examples/data/lo1.mps'
Reading started.
Using 'obj' as objective vector
Read 13 number of A nonzeros in 0.00 seconds.
Using 'rhs' as rhs vector
Using 'bound' as bound vector
Reading terminated. Time: 0.00
```

Read summary

```
  Type           : LO (linear optimization problem)
  Objective sense : max
  Scalar variables : 4
  Matrix variables : 0
  Constraints      : 3
  Cones           : 0
  Time            : 0.0
```

Problem

```
  Name           : lo1
```

(continues on next page)

(continued from previous page)

Objective sense : max
Type : LO (linear optimization problem)
Constraints : 3
Cones : 0
Scalar variables : 4
Matrix variables : 0
Integer variables : 0

Optimizer started.

Interior-point optimizer started.

Presolve started.

Linear dependency checker started.

Linear dependency checker terminated.

Eliminator started.

Freed constraints in eliminator : 0

Eliminator terminated.

Eliminator - tries : 1 time : 0.00

Lin. dep. - tries : 1 time : 0.00

Lin. dep. - number : 0

Presolve terminated. Time: 0.00

Optimizer - threads : 2

Optimizer - solved problem : the primal

Optimizer - Constraints : 3

Optimizer - Cones : 0

Optimizer - Scalar variables : 6 conic : 0

Optimizer - Semi-definite variables: 0 scalarized : 0

Factor - setup time : 0.00 dense det. time : 0.00

Factor - ML order time : 0.00 GP order time : 0.00

Factor - nonzeros before factor : 6 after factor : 6

Factor - dense dim. : 0 flops : 1.

↪ 06e+02

	ITE	PFEAS	DFEAS	GFEAS	PRSTATUS	POBJ	DOBJ	MU
--	-----	-------	-------	-------	----------	------	------	----

↪ TIME

0	8.0e+00	3.2e+00	3.5e+00	1.00e+00	1.000000000e+01	0.000000000e+00	1.0e+00
---	---------	---------	---------	----------	-----------------	-----------------	---------

↪ 0.01

1	4.2e+00	2.5e+00	4.7e-01	0.00e+00	3.093970927e+01	2.766058702e+01	2.6e+00
---	---------	---------	---------	----------	-----------------	-----------------	---------

↪ 0.01

2	4.2e-01	2.5e-01	4.6e-02	-1.82e-02	6.511676243e+01	6.308843559e+01	2.6e-01
---	---------	---------	---------	-----------	-----------------	-----------------	---------

↪ 0.01

3	3.6e-02	2.1e-02	3.9e-03	5.84e-01	8.096141239e+01	8.061962333e+01	2.2e-02
---	---------	---------	---------	----------	-----------------	-----------------	---------

↪ 0.01

4	1.5e-05	9.1e-06	1.7e-06	9.43e-01	8.333280389e+01	8.333241803e+01	9.2e-06
---	---------	---------	---------	----------	-----------------	-----------------	---------

↪ 0.01

5	1.5e-09	9.1e-10	1.7e-10	1.00e+00	8.333333328e+01	8.333333324e+01	9.2e-10
---	---------	---------	---------	----------	-----------------	-----------------	---------

↪ 0.01

Basis identification started.

Primal basis identification phase started.

ITER	TIME
------	------

0	0.00
---	------

Primal basis identification phase terminated. Time: 0.00

Dual basis identification phase started.

ITER	TIME
------	------

0	0.00
---	------

Dual basis identification phase terminated. Time: 0.00

Basis identification terminated. Time: 0.00

Interior-point optimizer terminated. Time: 0.01.

(continues on next page)

Optimizer terminated. Time: 0.02

Interior-point solution summary

Problem status : PRIMAL_AND_DUAL_FEASIBLE

Solution status : OPTIMAL

Primal. obj: 8.3333333280e+01 nrm: 5e+01 Viol. con: 1e-08 var: 0e+00

Dual. obj: 8.3333333242e+01 nrm: 4e+00 Viol. con: 2e-10 var: 5e-09

Basic solution summary

Problem status : PRIMAL_AND_DUAL_FEASIBLE

Solution status : OPTIMAL

Primal. obj: 8.3333333333e+01 nrm: 5e+01 Viol. con: 7e-15 var: 0e+00

Dual. obj: 8.3333333245e+01 nrm: 4e+00 Viol. con: 2e-10 var: 5e-09

Optimizer summary

Optimizer	-	time: 0.02
Interior-point	- iterations : 5	time: 0.01
Basis identification	-	time: 0.00
Primal	- iterations : 0	time: 0.00
Dual	- iterations : 0	time: 0.00
Clean primal	- iterations : 0	time: 0.00
Clean dual	- iterations : 0	time: 0.00
Simplex	-	time: 0.00
Primal simplex	- iterations : 0	time: 0.00
Dual simplex	- iterations : 0	time: 0.00
Mixed integer	- relaxations: 0	time: 0.00

Open file '/home/andrea/mosek/8/tools/examples/data/lo1.sol'

Start writing.

done writing. Time: 0.00

Open file '/home/andrea/mosek/8/tools/examples/data/lo1.bas'

Start writing.

done writing. Time: 0.00

Return code - 0 [MSK_RES_OK]

5.4 Solver Parameters

MOSEK comes with a large number of parameters that allows the user to tune the behavior of the optimizer. The typical settings which can be changed with solver parameters include:

- choice of the optimizer for linear problems,
- choice of primal/dual solver,
- turning presolve on/off,
- turning heuristics in the mixed-integer optimizer on/off,
- level of multi-threading,
- feasibility tolerances,
- solver termination criteria,
- behaviour of the license manager,

and more. All parameters have default settings which will be suitable for most typical users. Each parameter is identified by a unique string name and it can accept either integers or symbolic names, floating point values or symbolic strings. Please refer to [Sec. 11.2](#) for the complete list of available solver parameters.

5.4.1 Setting from command line

Setting solver parameters is possible using the command line option `-d`. If multiple parameters must be specified, option `-d` must be repeated for each one. For example, the next command will switch off presolve, set a feasibility tolerance and solve the problem from `lo1.opf`:

```
mosek -d MSK_IPAR_PRESOLVE_USE MSK_OFF -d MSK_DPAR_INTPNT_TOL_PFEAS 1.0e-8 lo1.opf
```

5.4.2 Using the Parameter File

Solver parameters can also be set using a parameter file, for example:

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

The syntax of the parameter file must obey a few simple rules:

- The file must begin with `BEGIN MOSEK` and end with `END MOSEK`.
- Empty lines and lines starting from a `%` sign are ignored.
- Each line contains a valid **MOSEK** parameter name followed by its value.

The parameter file can have any name. Assuming it has been called `mosek.par`, it can be used using the `-p` option as follows:

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

Command-line parameters override those from the parameter file in case of repetition. For instance

```
mosek -p mosek.par -d MSK_DPAR_INTPNT_TOL_PFEAS 1.0e-8 afiro.mps
```

will set `MSK_DPAR_INTPNT_TOL_PFEAS` to 10^{-8} using the value provided on the command line.

5.5 Command Line Arguments

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

- `-anapro`
Analyze the problem data.
- `-anasoli <name>`
Analyze the initial solution name e.g. `-anasoli bas`.
- `-anasolo <name>`
Analyze the final solution name e.g. `-anasolo itg`.
- `-a`
MOSEK is started in AMPL mode.
- `-basi <name>`
Input basic solution file name.
- `-baso <name>`
Output basic solution file name.
- `-d <name> <value>`
Define the value `value` for the **MOSEK** parameter `name`.

-dbgmem <name>
 Name of memory debug file.

-f
 Complete license information is printed.

-h, -?
 Help.

-inti <name>
 Input integer solution file name.

-into <name>
 Output integer solution file name.

-itri <name>
 Input interior point solution file name.

-itro <name>
 Output interior point solution file name.

-info <name>
 Infeasible subproblem output file name.

-infrepo <name>
 Feasibility reparation output file.

-jsoli <name>
 Input JSON format solution file name.

-jsolo <name>
 Output JSON format solution file name.

-l, -L <dir>
 dir is the directory where the **MOSEK** license file `mosek.lic` is located.

-max
 The problem is maximized.

-min
 The problem is minimized.

-n
 Ignore errors in subsequent parameter settings.

-optserv <url>
 Use an OptServer specified by an URL of the form `http://host:port`.

-out <name>
 Write the task to a data file named `name`. See [Sec. 12](#).

-p <name>, **-pari** <name>
 Name of the input parameter file.

-paro <name>
 Name of the output parameter file.

-primalrepair
 Repair a primal infeasible problem. See [Sec. 10.2](#).

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

-removeitg
 Removes all integer constraints after reading the problem.

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

-q <name>
 Name of an optional log file.

-sen <file>
 Perform sensitivity analysis based on file.

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

-toconic
 Translate to conic form after reading.

-v
MOSEK version is printed and no optimization is performed.

-w
 If this options is on, then **MOSEK** will wait for a license.

- x Do not run the optimizer. Useful for converting between file formats.
- List all possible solver parameters with default value, lower bound and upper bound (if applicable).

5.6 The license system

MOSEK is a commercial product that **always** needs a valid license to work. **MOSEK** uses a third party license manager to implement license checking. The number of license tokens provided determines the number of optimizations that can be run simultaneously.

By default a license token remains checked out for the whole execution of the command line tool. If the license is not unlimited, then the number of tokens determines the maximal number of processes that can run simultaneously. In this case setting the license wait flag with the parameter *MSK_IPAR_LICENSE_WAIT* will force **MOSEK** to wait until a license token becomes available instead of returning with an error.

Chapter 6

The MOSEK-bundled AMPL shell

AMPL is a modeling language for specifying linear and nonlinear optimization models in a natural way. AMPL also makes it easy to solve the problem and e.g. display the solution or part of it. We will not discuss the specifics of the AMPL language here but instead refer the reader to [FGK03], <http://ampl.com/BOOK/download.html> and the AMPL website <http://www.ampl.com>.

AMPL cannot solve optimization problems by itself but requires a link to an optimizer. The **MOSEK** distribution includes an AMPL link which makes it possible to use **MOSEK** as an optimizer within AMPL. The link can be used from any AMPL shell.

Note:

- To use **MOSEK** from AMPL you need to set up the system path to the **MOSEK** command line tool.
- It is possible to specify problems in AMPL that cannot be solved by **MOSEK**. The optimization problem must be a smooth convex optimization problem as discussed in [Sec. 8](#).

We refer to your AMPL documentation for how to specify optimization problems in AMPL and attach **MOSEK** as a solver.

6.1 An example

In many instances, you can successfully apply **MOSEK** simply by specifying the model and data, setting the solver option to **MOSEK**, and typing `solve`.

Consider a simple linear optimization problem formulated as an AMPL model in [Listing 6.1](#).

Listing 6.1: An example of an optimization problem in AMPL language.

```
set NUTR ordered;
set FOOD ordered;

param cost {FOOD} >= 0;
param f_min {FOOD} >= 0, default 0;
param f_max {j in FOOD} >= f_min[j], default Infinity;

param n_min {NUTR} >= 0, default 0;
param n_max {i in NUTR} >= n_min[i], default Infinity;

param amt {NUTR,FOOD} >= 0;

# -----
```

(continues on next page)

(continued from previous page)

```
var Buy {j in FOOD} >= f_min[j], <= f_max[j];

# -----

minimize Total_Cost: sum {j in FOOD} cost[j] * Buy[j];

minimize Nutr_Amt {i in NUTR}: sum {j in FOOD} amt[i,j] * Buy[j];

# -----

subject to Diet {i in NUTR}:
    n_min[i] <= sum {j in FOOD} amt[i,j] * Buy[j] <= n_max[i];
```

We can specify the input data using an input file again following the AMPL syntax, as in [Listing 6.2](#).

Listing 6.2: An example of data for an optimization problem using AMPL language.

```
param:  FOOD:                cost  f_min  f_max :=
"Quarter Pounder w/ Cheese"  1.84    .    .
"McLean Deluxe w/ Cheese"   2.19    .    .
"Big Mac"                    1.84    .    .
"Filet-O-Fish"              1.44    .    .
"McGrilled Chicken"         2.29    .    .
"Fries, small"              .77     .    .
"Sausage McMuffin"          1.29    .    .
"1% Lowfat Milk"            .60     .    .
"Orange Juice"              .72     .    . ;

param:  NUTR:    n_min  n_max :=
    Cal      2000    .
    Carbo    350    375
    Protein   55     .
    VitA     100     .
    VitC     100     .
    Calc     100     .
    Iron     100     . ;

param amt (tr):
                                Cal  Carbo Protein  VitA  VitC  Calc  Iron :=
"Quarter Pounder w/ Cheese"  510   34   28   15   6   30   20
"McLean Deluxe w/ Cheese"   370   35   24   15   10  20   20
"Big Mac"                   500   42   25   6    2   25   20
"Filet-O-Fish"              370   38   14   2    0   15   10
"McGrilled Chicken"         400   42   31   8    15  15   8
"Fries, small"              220   26   3    0    15   0   2
"Sausage McMuffin"          345   27   15   4    0   20  15
"1% Lowfat Milk"            110   12   9    10   4   30   0
"Orange Juice"              80    20   1    2   120  2   2 ;
```

In AMPL type in the commands:

```
ampl: model diet.mod;
ampl: data diet.dat;
ampl: option solver mosek;
ampl: solve;
```

The resulting output is:

```

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

Objective = Total_Cost

```

6.2 Retrieving solutions

6.2.1 Status codes

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

```
ampl: display solve_result_num
```

Please refer to table [Table 6.1](#) for possible values of this parameter.

Table 6.1: Interpretation of `solve_result_num`.

Value	Message
0	the solution is optimal.
100	suboptimal primal solution.
101	superoptimal (dual feasible) solution.
150	the solution is near optimal.
200	primal infeasible problem.
300	dual infeasible problem.
400	too many iterations.
500	solution status is unknown.
501	ill-posed problem, solution status is unknown.
> 501	Mapped MOSEK response code. See note below.

MOSEK response codes are mapped to AMPL return codes greater than 501. In order to get the actual response code the base value 501 must be subtracted. For example: the AMPL return code 502 corresponds to **MOSEK** response code 1.

6.2.2 Which solution is returned

MOSEK can produce three types of solutions: basic, interior point and integer. The solution returned to AMPL is determined according to the following rules:

- For problems containing integer variables only the integer solution is available and it is returned.
- For nonlinear problems only the interior point solution is available and it is returned.
- For linear problems, if both basic and interior point solution are available, then the basic solution is returned. Otherwise the only available solution is returned.

6.3 Optimizer options

6.3.1 The MOSEK parameter database

The **MOSEK** optimizer can be controller using solver parameters, as described in [Sec. 5.4](#). These parameters can be modified within AMPL as shown in the example below:

```
ampl: model diet.mod;
ampl: data diet.dat;
ampl: option solver mosek;
ampl: option mosek_options
ampl? 'msk_ipar_optimizer = msk_optimizer_primal_simplex \
ampl? msk_ipar_sim_max_iterations = 100000';
ampl: solve;
```

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

```
parameter_name = value
```

where `parameter_name` is a valid **MOSEK** parameter name. See [Sec. 11.2](#) for a description of all valid **MOSEK** parameters.

An alternative way of specifying the parameters is

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

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

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

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

6.3.2 Options

MOSEK recognizes the following AMPL options.

outlev

Controls the amount of printed output. 0 means no printed output and a higher value means progressively more output. An example of setting `outlev` is as follows:

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

wantsol

Controls the solution information generated when run in standalone mode (called without the argument `-AMPL`). It should be constructed as the sum of

1	to write a <code>.sol</code> file
2	to print the primal variable values
4	to print the dual variable values
8	to suppress printing the solution message

We refer the reader to the AMPL manual [\[FGK03\]](#) for more details.

6.3.3 Passing variable names to MOSEK

AMPL assigns meaningful names to all the constraints and variables. Since **MOSEK** uses item names in error and log messages, it may be useful to pass the AMPL names to **MOSEK**. This can be achieved with the command:

```
ampl: option auxfiles rc;
ampl: solve;
```

6.4 Hot-start

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

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

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

The resulting output is:

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

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

Optimizer started.
Simplex optimizer started.
Presolve started.
Linear dependency checker started.
Linear dependency checker terminated.
Presolve  - Stk. size (kb) : 0
Eliminator - tries          : 0                time                : 0.00
Eliminator - elim's         : 0
Lin. dep.  - tries          : 1                time                : 0.00
Lin. dep.  - number         : 0
Presolve terminated. Time: 0.00
Primal simplex optimizer started.
Primal simplex optimizer setup started.
Primal simplex optimizer setup terminated.
Optimizer  - solved problem      : the primal
Optimizer  - constraints         : 7                variables          : 9
Optimizer  - hotstart           : no
ITER       DEGITER(%)  PFEAS      DFEAS      POBJ      DOBJ
↪          TIME       TOTTIME
(continues on next page)
```

(continued from previous page)

```
0          0.00          1.40e+03    NA          1.2586666667e+01    NA
↪          0.00          0.01
3          0.00          0.00e+00    NA          1.4855737705e+01    NA
↪          0.00          0.01
Primal simplex optimizer terminated.
Simplex optimizer terminated. Time: 0.00.
Optimizer terminated. Time: 0.01
Return code - 0 [MSK_RES_OK]
MOSEK finished.
Problem status      : PRIMAL_AND_DUAL_FEASIBLE
Solution status     : OPTIMAL
Primal objective    : 14.8557377
Dual objective      : 14.8557377

Objective = Total_Cost
Buy.sstatus [*] :=
'Quarter Pounder w/ Cheese' bas
'McLean Deluxe w/ Cheese' low
'Big Mac' low
Filet-O-Fish low
'McGrilled Chicken' low
'Fries, small' bas
'Sausage McMuffin' low
'1% Lowfat Milk' bas
'Orange Juice' low
;
Accepted: msk_ipar_optimizer          = MSK_OPTIMIZER_PRIMAL_SIMPLEX
Accepted: outlev                      = 2
Basic solution
Problem status  : UNKNOWN
Solution status : UNKNOWN
Primal - objective: 1.4855737705e+01   eq. infeas.: 3.97e+03 max bound infeas.: 2.
↪00e+03
Dual - objective: 0.0000000000e+00   eq. infeas.: 7.14e-01 max bound infeas.: 0.
↪00e+00

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

(continues on next page)

(continued from previous page)

```
Optimizer - constraints          : 7          variables          : 9
Optimizer - hotstart            : yes
Optimizer - Num. basic          : 7          Basis rank          : 7
Optimizer - Valid bas. fac.     : no
ITER      DEGITER(%)  PFEAS      DFEAS      POBJ          DOBJ          ┐
↪  TIME      TOTTIME
0          0.00      0.00e+00    NA          1.4855737705e+01    NA          ┐
↪  0.00      0.01
0          0.00      0.00e+00    NA          1.4855737705e+01    NA          ┐
↪  0.00      0.01
Primal simplex optimizer terminated.
Simplex optimizer terminated. Time: 0.00.
Optimizer terminated. Time: 0.01
Return code - 0 [MSK_RES_OK]
MOSEK finished.
Problem status      : PRIMAL_AND_DUAL_FEASIBLE
Solution status     : OPTIMAL
Primal objective    : 14.8557377
Dual objective      : 14.8557377

Objective = Total_Cost
```

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

6.5 Infeasibility report

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

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

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

For further details about infeasibility report see [Sec. 10.2](#).

6.6 Sensitivity analysis

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

```
sensitivity = 1
```

Results are returned in variable/constraint suffixes as follows:

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

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

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

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

(continues on next page)

(continued from previous page)

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

The resulting output is:

```
Return code - 0 [MSK_RES_OK]
MOSEK finished.
Problem status : PRIMAL_AND_DUAL_FEASIBLE
Solution status : OPTIMAL
Primal objective : 14.8557377
Dual objective : 14.8557377

suffix up OUT;
suffix down OUT;
suffix current OUT;
Objective = Total_Cost
:  _var.down _var.current      _var.up      _var      :=
1  1.37385    1.84             1.86075    4.38525
2  1.8677     2.19             Infinity    0
3  1.82085    1.84             Infinity    0
4  1.35466    1.44             Infinity    0
5  1.57633    2.29             Infinity    0
6  0.094      0.77             0.794851    6.14754
7  1.22759    1.29             Infinity    0
8  0.57559    0.6              0.910769    3.42213
9  0.657279   0.72             Infinity    0
;
ampl: display _con.down,_con.current,_con.up,_con;
:  _con.down  _con.current  _con.up  _con      :=
1  -Infinity  2000          3965.37  0
2           297.6         350         375     0.0277049
3  -Infinity  55           172.029  0
4           63.0531       100         195.388  0.0267541
5  -Infinity  100          132.213  0
6  -Infinity  100          234.221  0
7           17.6923       100         142.821  0.0248361
;
```

6.7 Using the command line version of the AMPL interface

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

When the problem has been loaded into AMPL, the commands

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

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

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

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

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

The option *-a* indicates that **MOSEK** is invoked in AMPL mode. When **MOSEK** is invoked in AMPL mode the standard **MOSEK** command line options should appear *after* the *-a* option except for the file name which should be the first argument. As the above example demonstrates **MOSEK** accepts command line options following the AMPL convention. To see which command line arguments **MOSEK** accepts in AMPL mode write:

```
mosek -= -a
```

For linear, quadratic and quadratically constrained problems a text file representation of the problem can be obtained by performing one of the following conversions:

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

Chapter 7

Debugging Tutorials

This collection of tutorials contains basic techniques for debugging optimization problems using tools available in **MOSEK**: optimizer log, solution summary, infeasibility report, command-line tools. It is intended as a first line of technical help for issues such as: Why do I get solution status *unknown* and how can I fix it? Why is my model infeasible while it shouldn't be? Should I change some parameters? Can the model solve faster? etc.

The major steps when debugging a model are always:

- Consult the log output.
- Run the optimization and analyze the log output, see [Sec. 7.1](#). In particular:
 - check if the problem setup (number of constraints/variables etc.) matches your expectation.
 - check solution summary and solution status.
- Dump the problem to disk if necessary to continue analysis.
 - use a human-readable text format, preferably `*.ptf` if you want to check the problem structure by hand. Assign names to variables and constraints to make them easier to identify.
 - use the **MOSEK** native format `*.task.gz` when submitting a bug report or support question.
- Fix problem setup, improve the model, locate infeasibility or adjust parameters, depending on the diagnosis.

See the following sections for details.

7.1 Understanding optimizer log

The optimizer produces a log which splits roughly into four sections:

1. summary of the input data,
2. presolve and other pre-optimize problem setup stages,
3. actual optimizer iterations,
4. solution summary.

In this tutorial we show how to analyze the most important parts of the log when initially debugging a model: input data (1) and solution summary (4). For the iterations log (3) see [Sec. 9.3.4](#) or [Sec. 9.4.4](#).

7.1.1 Input data

If **MOSEK** behaves very far from expectations it may be due to errors in problem setup. The log file will begin with a summary of the structure of the problem, which looks for instance like:

```

Problem
  Name           :
  Objective sense : minimize
  Type           : CONIC (conic optimization problem)
  Constraints     : 234
  Affine conic cons. : 5348
  Disjunctive cons. : 0
  Cones          : 0
  Scalar variables : 20693
  Matrix variables : 0
  Integer variables : 0

```

This can be consulted to eliminate simple errors: wrong objective sense, wrong number of variables etc. Note that some modeling tools can introduce additional variables and constraints to the model and perturb the model even further (such as by dualizing). In most **MOSEK** APIs the problem dimensions should match exactly what the user specified.

If this is not sufficient a bit more information can be obtained by dumping the problem to a file (see [Sec. 7](#)) and using the `anapro` option of any of the command line tools. This will produce a longer summary similar to:

```

** Variables
scalar: 20414      integer: 0      matrix: 0
low: 2082          up: 5014        ranged: 0      free: 12892    fixed: 426

** Constraints
all: 20413
low: 10028        up: 0           ranged: 0      free: 0        fixed: 10385

** Affine conic constraints (ACC)
QUAD: 1           dims: 2865: 1
RQUAD: 2507       dims: 3: 2507

** Problem data (numerics)
|c|               nnz: 10028      min=2.09e-05   max=1.00e+00
|A|               nnz: 597023     min=1.17e-10   max=1.00e+00
blx               fin: 2508       min=-3.60e+09   max=2.75e+05
bux               fin: 5440       min=0.00e+00   max=2.94e+08
blc               fin: 20413     min=-7.61e+05   max=7.61e+05
buc               fin: 10385     min=-5.00e-01   max=0.00e+00
|F|               nnz: 612301     min=8.29e-06   max=9.31e+01
|g|               nnz: 1203       min=5.00e-03   max=1.00e+00

```

Again, this can be used to detect simple errors, such as:

- Wrong type of conic constraint was used or it has wrong dimension.
- The bounds for variables or constraints are incorrect or incomplete.
- The model is otherwise incomplete.
- Suspicious values of coefficients.
- For various data sizes the model does not scale as expected.

Finally saving the problem in a human-friendly text format such as LP or PTF (see [Sec. 7](#)) and analyzing it by hand can reveal if the model is correct.

Warnings and errors

At this stage the user can encounter warnings which should not be ignored, unless they are well-understood. They can also serve as hints as to numerical issues with the problem data. A typical warning of this kind is

```
MOSEK warning 53: A numerically large upper bound value 2.9e+08 is specified for
↪variable 'absh[107]' (2613).
```

Warnings do not stop the problem setup. If, on the other hand, an error occurs then the model will become invalid. The user should make sure to test for errors/exceptions from all API calls that set up the problem and validate the data.

7.1.2 Solution summary

The last item in the log is the solution summary.

Continuous problem

Optimal solution

A typical solution summary for a continuous (linear, conic, quadratic) problem looks like:

```
Problem status : PRIMAL_AND_DUAL_FEASIBLE
Solution status : OPTIMAL
Primal.  obj: 8.7560516107e+01    nrm: 1e+02    Viol.  con: 3e-12    var: 0e+00    ↪
↪acc: 3e-11
Dual.    obj: 8.7560521345e+01    nrm: 1e+00    Viol.  con: 5e-09    var: 9e-11    ↪
↪acc: 0e+00
```

It contains the following elements:

- Problem and solution status.
- A summary of the primal solution: objective value, infinity norm of the solution vector and maximal violations of variables and constraints of different types. The violation of a linear constraint such as $a^T x \leq b$ is $\max(a^T x - b, 0)$. The violation of a conic constraint is the distance to the cone.
- The same for the dual solution.

The features of the solution summary which characterize a very good and accurate solution and a well-posed model are:

- **Status:** The solution status is `OPTIMAL`.
- **Duality gap:** The primal and dual objective values are (almost) identical, which proves the solution is (almost) optimal.
- **Norms:** Ideally the norms of the solution and the objective values should not be too large. This of course depends on the input data, but a huge solution norm can be an indicator of issues with the scaling, conditioning and/or well-posedness of the model. It may also indicate that the problem is borderline between feasibility and infeasibility and sensitive to small perturbations in this respect.
- **Violations:** The violations are close to zero, which proves the solution is (almost) feasible. Observe that due to rounding errors it can be expected that the violations are proportional to the norm (`nrm:`) of the solution. It is rarely the case that violations are exactly zero.

Solution status UNKNOWN

A typical example with solution status UNKNOWN due to numerical problems will look like:

```
Problem status : UNKNOWN
Solution status : UNKNOWN
Primal.  obj: 1.3821656824e+01    nrm: 1e+01    Viol.  con: 2e-03    var: 0e+00    ⌋
↪acc: 0e+00
Dual.    obj: 3.0119004098e-01    nrm: 5e+07    Viol.  con: 4e-16    var: 1e-01    ⌋
↪acc: 0e+00
```

Note that:

- The primal and dual objective are very different.
- The dual solution has very large norm.
- There are considerable violations so the solution is likely far from feasible.

Follow the hints in [Sec. 7.2](#) to resolve the issue.

Solution status UNKNOWN with a potentially useful solution

Solution status UNKNOWN does not necessarily mean that the solution is completely useless. It only means that the solver was unable to make any more progress due to numerical difficulties, and it was not able to reach the accuracy required by the termination criteria (see [Sec. 9.3.2](#)). Consider for instance:

```
Problem status : UNKNOWN
Solution status : UNKNOWN
Primal.  obj: 3.4531019648e+04    nrm: 1e+05    Viol.  con: 7e-02    var: 0e+00    ⌋
↪acc: 0e+00
Dual.    obj: 3.4529720645e+04    nrm: 8e+03    Viol.  con: 1e-04    var: 2e-04    ⌋
↪acc: 0e+00
```

Such a solution may still be useful, and it is always up to the user to decide. It may be a good enough approximation of the optimal point. For example, the large constraint violation may be due to the fact that one constraint contained a huge coefficient.

Infeasibility certificate

A primal infeasibility certificate is stored in the dual variables:

```
Problem status : PRIMAL_INFEASIBLE
Solution status : PRIMAL_INFEASIBLE_CER
Dual.    obj: 2.9238975853e+02    nrm: 6e+02    Viol.  con: 0e+00    var: 1e-11    ⌋
↪acc: 0e+00
```

It is a Farkas-type certificate as described in [Sec. 8.2.2](#). In particular, for a good certificate:

- The dual objective is positive for a minimization problem, negative for a maximization problem. Ideally it is well bounded away from zero.
- The norm is not too big and the violations are small (as for a solution).

If the model was not expected to be infeasible, the likely cause is an error in the problem formulation. Use the hints in [Sec. 7.1.1](#) and [Sec. 7.3](#) to locate the issue.

Just like a solution, the infeasibility certificate can be of better or worse quality. The infeasibility certificate above is very solid. However, there can be less clear-cut cases, such as for example:

```
Problem status : PRIMAL_INFEASIBLE
Solution status : PRIMAL_INFEASIBLE_CER
Dual.    obj: 1.6378689238e-06    nrm: 6e+05    Viol.  con: 7e-03    var: 2e-04    ⌋
↪acc: 0e+00
```

This infeasibility certificate is more dubious because the dual objective is positive, but barely so in comparison with the large violations. It also has rather large norm. This is more likely an indication that the problem is borderline between feasibility and infeasibility or simply ill-posed and sensitive to tiny variations in input data. See [Sec. 7.3](#) and [Sec. 7.2](#).

The same remarks apply to dual infeasibility (i.e. unboundedness) certificates. Here the primal objective should be negative a minimization problem and positive for a maximization problem.

7.1.3 Mixed-integer problem

Optimal integer solution

For a mixed-integer problem there is no dual solution and a typical optimal solution report will look as follows:

```
Problem status : PRIMAL_FEASIBLE
Solution status : INTEGER_OPTIMAL
Primal.  obj: 6.0111122960e+06    nrm: 1e+03    Viol.  con: 2e-13    var: 2e-14    ̐
↪itg: 5e-15
```

The interpretation of all elements is as for a continuous problem. The additional field `itg` denotes the maximum violation of an integer variable from being an exact integer.

Feasible integer solution

If the solver found an integer solution but did not prove optimality, for instance because of a time limit, the solution status will be `PRIMAL_FEASIBLE`:

```
Problem status : PRIMAL_FEASIBLE
Solution status : PRIMAL_FEASIBLE
Primal.  obj: 6.0114607792e+06    nrm: 1e+03    Viol.  con: 2e-13    var: 2e-13    ̐
↪itg: 4e-15
```

In this case it is valuable to go back to the optimizer summary to see how good the best solution is:

```
31      35      1      0      6.0114607792e+06      6.0078960892e+06      0.06    ̐
↪      4.1

Objective of best integer solution : 6.011460779193e+06
Best objective bound                : 6.007896089225e+06
```

In this case the best integer solution found has objective value `6.011460779193e+06`, the best proved lower bound is `6.007896089225e+06` and so the solution is guaranteed to be within 0.06% from optimum. The same data can be obtained as information items through an API. See also [Sec. 9.4](#) for more details.

Infeasible problem

If the problem is declared infeasible the summary is simply

```
Problem status : PRIMAL_INFEASIBLE
Solution status : UNKNOWN
Primal.  obj: 0.0000000000e+00    nrm: 0e+00    Viol.  con: 0e+00    var: 0e+00    ̐
↪itg: 0e+00
```

If infeasibility was not expected, consult [Sec. 7.3](#).

7.2 Addressing numerical issues

The suggestions in this section should help diagnose and solve issues with numerical instability, in particular **UNKNOWN** solution status or solutions with large violations. Since numerically stable models tend to solve faster, following these hints can also dramatically shorten solution times.

We always recommend that issues of this kind are addressed by reformulating or rescaling the model, since it is the modeler who has the best insight into the structure of the problem and can fix the cause of the issue.

7.2.1 Formulating problems

Scaling

Make sure that all the data in the problem are of comparable orders of magnitude. This applies especially to the linear constraint matrix. Use [Sec. 7.1.1](#) if necessary. For example a report such as

A	nnz: 597023	min=1.17e-6	max=2.21e+5
---	-------------	-------------	-------------

means that the ratio of largest to smallest elements in **A** is 10^{11} . In this case the user should rescale or reformulate the model to avoid such spread which makes it difficult for **MOSEK** to scale the problem internally. In many cases it may be possible to change the units, i.e. express the model in terms of rescaled variables (for instance work with millions of dollars instead of dollars, etc.).

Similarly, if the objective contains very different coefficients, say

$$\text{maximize } 10^{10}x + y$$

then it is likely to lead to inaccuracies. The objective will be dominated by the contribution from x and y will become insignificant.

Removing huge bounds

Never use a very large number as replacement for ∞ . Instead define the variable or constraint as unbounded from below/above. Similarly, avoid artificial huge bounds if you expect they will not become tight in the optimal solution.

Avoiding linear dependencies

As much as possible try to avoid linear dependencies and near-linear dependencies in the model. See [Example 7.3](#).

Avoiding ill-posedness

Avoid continuous models which are ill-posed: the solution space is degenerate, for example consists of a single point (technically, the Slater condition is not satisfied). In general, this refers to problems which are borderline between feasible and infeasible. See [Example 7.1](#).

Scaling the expected solution

Try to formulate the problem in such a way that the expected solution (both primal and dual) is not very large. Consult the solution summary [Sec. 7.1.2](#) to check the objective values or solution norms.

7.2.2 Further suggestions

Here are other simple suggestions that can help locate the cause of the issues. They can also be used as hints for how to tune the optimizer if fixing the root causes of the issue is not possible.

- Remove the objective and solve the feasibility problem. This can reveal issues with the objective.
- Change the objective or change the objective sense from minimization to maximization (if applicable). If the two objective values are almost identical, this may indicate that the feasible set is very small, possibly degenerate.
- Perturb the data, for instance bounds, very slightly, and compare the results.
- For linear problems: solve the problem using a different optimizer by setting the parameter `MSK_IPAR_OPTIMIZER` and compare the results.
- Force the optimizer to solve the primal/dual versions of the problem by setting the parameter `MSK_IPAR_INTPNT_SOLVE_FORM` or `MSK_IPAR_SIM_SOLVE_FORM`. **MOSEK** has a heuristic to decide whether to dualize, but for some problems the guess is wrong an explicit choice may give better results.
- Solve the problem without presolve or some of its parts by setting the parameter `MSK_IPAR_PRESOLVE_USE`, see Sec. 9.1.
- Use different numbers of threads (`MSK_IPAR_NUM_THREADS`) and compare the results. Very different results indicate numerical issues resulting from round-off errors.

If the problem was dumped to a file, experimenting with various parameters is facilitated with the **MOSEK** Command Line Tool or **MOSEK** Python Console Sec. 7.4.

7.2.3 Typical pitfalls

Example 7.1 (Ill-posedness). A toy example of this situation is the feasibility problem

$$(x - 1)^2 \leq 1, (x + 1)^2 \leq 1$$

whose only solution is $x = 0$ and moreover replacing any 1 on the right hand side by $1 - \varepsilon$ makes the problem infeasible and replacing it by $1 + \varepsilon$ yields a problem whose solution set is an interval (fully-dimensional). This is an example of ill-posedness.

Example 7.2 (Huge solution). If the norm of the expected solution is very large it may lead to numerical issues or infeasibility. For example the problem

$$(10^{-4}, x, 10^3) \in \mathcal{Q}_r^3$$

may be declared infeasible because the expected solution must satisfy $x \geq 5 \cdot 10^9$.

Example 7.3 (Near linear dependency). Consider the following problem:

$$\begin{array}{llllll} \text{minimize} & & & & & \\ \text{subject to} & x_1 & + & x_2 & & = & 1, \\ & & & & x_3 & + & x_4 & = & 1, \\ & - & x_1 & & - & x_3 & & = & -1 + \varepsilon, \\ & & - & x_2 & & - & x_4 & = & -1, \\ & x_1, & & x_2, & & x_3, & & x_4 & \geq & 0. \end{array}$$

If we add the equalities together we obtain:

$$0 = \varepsilon$$

which is infeasible for any $\varepsilon \neq 0$. Here infeasibility is caused by a linear dependency in the constraint matrix coupled with a precision error represented by the ε . Indeed if a problem contains linear dependencies then the problem is either infeasible or contains redundant constraints. In the above case any of the equality constraints can be removed while not changing the set of feasible solutions. To summarize linear dependencies in the constraints can give rise to infeasible problems and therefore it is better to avoid them.

Example 7.4 (Presolving very tight bounds). Next consider the problem

$$\begin{array}{ll} \text{minimize} & \\ \text{subject to} & x_1 - 0.01x_2 = 0, \\ & x_2 - 0.01x_3 = 0, \\ & x_3 - 0.01x_4 = 0, \\ & x_1 \geq -10^{-9}, \\ & x_1 \leq 10^{-9}, \\ & x_4 \geq 10^{-4}. \end{array}$$

Now the **MOSEK** presolve will, for the sake of efficiency, fix variables (and constraints) that have tight bounds where tightness is controlled by the parameter `MSK_DPAR_PRESOLVE_TOL_X`. Since the bounds

$$-10^{-9} \leq x_1 \leq 10^{-9}$$

are tight, presolve will set $x_1 = 0$. It easy to see that this implies $x_4 = 0$, which leads to the incorrect conclusion that the problem is infeasible. However a tiny change of the value 10^{-9} makes the problem feasible. In general it is recommended to avoid ill-posed problems, but if that is not possible then one solution is to reduce parameters such as `MSK_DPAR_PRESOLVE_TOL_X` to say 10^{-10} . This will at least make sure that presolve does not make the undesired reduction.

7.3 Debugging infeasibility

When solving an optimization problem one typically expects to get an optimal solution, but in some cases, either by design, or (most frequently) due to an error in the formulation, the problem may become infeasible (have no solution at all).

This section

- describes the intuitions behind infeasibility,
- helps to debug (unexpectedly) infeasible problems using the command line tool and by inspecting infeasibility reports and problem data by hand,
- gives some hints for how to modify the formulation to identify the reasons for infeasibility.

An infeasibility certificate is only available for continuous problems, however the hints in [Sec. 7.3.4](#) apply to a large extent also to mixed-integer problems.

7.3.1 Numerical issues

Infeasible problem status may be just an artifact of numerical issues appearing when the problem is badly-scaled, barely feasible or otherwise ill-conditioned so that it is unstable under small perturbations of the data or round-off errors. This may be visible in the solution summary if the infeasibility certificate has poor quality. See [Sec. 7.1.2](#) for how to diagnose that and [Sec. 7.2](#) for possible hints. [Sec. 7.2.3](#) contains examples of situations which may lead to infeasibility for numerical reasons.

We refer to [Sec. 7.2](#) for further information on dealing with those sort of issues. For the rest of this section we concentrate on the case when the solution summary leaves little doubt that the problem solved by the optimizer actually is infeasible.

7.3.2 Locating primal infeasibility

As an example of a primal infeasible problem consider 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 Fig. 7.1.

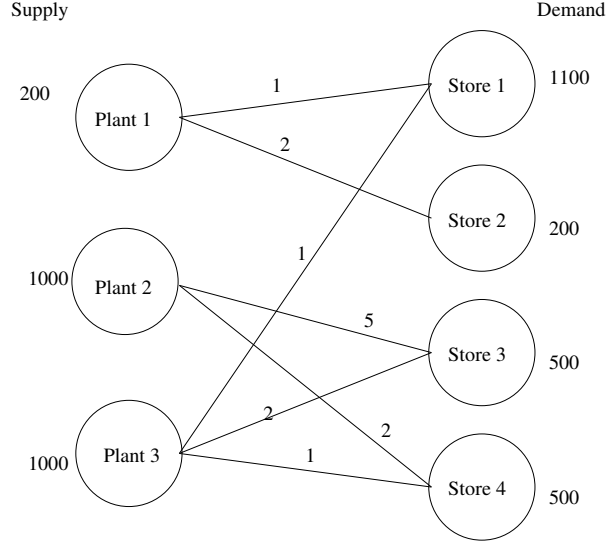


Fig. 7.1: Supply, demand and cost of transportation.

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

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

exceeds the total supply

$$2200 = 200 + 1000 + 1000$$

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

$$\begin{aligned}
& \text{minimize} && x_{11} + 2x_{12} + 5x_{23} + 2x_{24} + x_{31} + 2x_{33} + x_{34} \\
& \text{subject to} && s_0 : x_{11} + x_{12} \leq 200, \\
& && s_1 : x_{23} + x_{24} \leq 1000, \\
& && s_2 : x_{31} + x_{33} + x_{34} \leq 1000, \\
& && d_1 : x_{11} + x_{31} = 1100, \\
& && d_2 : x_{12} = 200, \\
& && d_3 : x_{23} + x_{33} = 500, \\
& && d_4 : x_{24} + x_{34} = 500, \\
& && x_{ij} \geq 0.
\end{aligned} \tag{7.1}$$

Solving problem (7.1) using **MOSEK** will result in an infeasibility status. The infeasibility certificate is contained in the dual variables and can be accessed from an API. The variables and constraints with nonzero solution values form an infeasible subproblem, which frequently is very small. See Sec. 8.2.1 or Sec. 8.2.2 for detailed specifications of infeasibility certificates.

A short infeasibility report can also be printed to the log stream. It can be turned on by setting the parameter `MSK_IPAR_INFEAS_REPORT_AUTO` to `MSK_ON`. This causes **MOSEK** to print a report on variables and constraints which are involved in infeasibility in the above sense, i.e. have nonzero values in the certificate. The parameter `MSK_IPAR_INFEAS_REPORT_LEVEL` controls the amount of information presented in the infeasibility report. The default value is 1. For the above example the report is

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 corresponding to constraints and variables. It is a selection of those lines from the problem solution which are important in understanding primal infeasibility. In this case the constraints s0, s2, d1, d2 and variables x33, x34 are of importance because of nonzero dual values. The columns Dual lower and Dual upper contain the values of dual variables s_l^c , s_u^c , s_l^x and s_u^x in the primal infeasibility certificate (see Sec. 8.2.1).

In our example the certificate means that an appropriate linear combination of constraints s0, s1 with coefficient $s_u^c = 1$, constraints d1 and d2 with coefficient $s_u^c - s_l^c = 0 - 1 = -1$ and lower bounds on x33 and x34 with coefficient $-s_l^x = -1$ gives a contradiction. Indeed, the combination of the four involved constraints is $x_{33} + x_{34} \leq -100$ (as indicated in the introduction, the difference between supply and demand).

It is also possible to extract the infeasible subproblem with the command-line tool. For an infeasible problem called infeas.lp the command:

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

will produce the file rinfeas.bas.inf.lp which contains the infeasible subproblem. Because of its size it may be easier to work with than the original problem file.

Returning to the transportation example, we discover that removing the fifth constraint $x_{12} = 200$ makes the problem feasible. Almost all undesired infeasibilities should be fixable at the modeling stage.

7.3.3 Locating dual infeasibility

A problem may also be *dual infeasible*. In this case the primal problem is usually unbounded, meaning that feasible solutions exists such that the objective tends towards infinity. For example, consider the problem

$$\begin{aligned}
 &\text{maximize} && 200y_1 + 1000y_2 + 1000y_3 + 1100y_4 + 200y_5 + 500y_6 + 500y_7 \\
 &\text{subject to} && y_1 + y_4 \leq 1, \quad y_1 + y_5 \leq 2, \quad y_2 + y_6 \leq 5, \quad y_2 + y_7 \leq 2, \\
 &&& y_3 + y_4 \leq 1, \quad y_3 + y_6 \leq 2, \quad y_3 + y_7 \leq 1 \\
 &&& y_1, y_2, y_3 \leq 0
 \end{aligned}$$

which is dual to (7.1) (and therefore is dual infeasible). The dual infeasibility report may look as follows:

MOSEK DUAL INFEASIBILITY REPORT.

Problem status: The problem is dual infeasible

The following constraints are involved in the infeasibility.

Index	Name	Activity	Objective	Lower bound	Upper bound
↪ bound					

(continues on next page)

(continued from previous page)

5	x33	-1.000000e+00	NONE	2.
	↪000000e+00			
6	x34	-1.000000e+00	NONE	1.
	↪000000e+00			
The following variables are involved in the infeasibility.				
Index	Name	Activity	Objective	Lower bound Upper bound
0	y1	-1.000000e+00	2.000000e+02	NONE 0.
	↪000000e+00			
2	y3	-1.000000e+00	1.000000e+03	NONE 0.
	↪000000e+00			
3	y4	1.000000e+00	1.100000e+03	NONE NONE
4	y5	1.000000e+00	2.000000e+02	NONE NONE
Interior-point solution summary				
Problem status : DUAL_INFEASIBLE				
Solution status : DUAL_INFEASIBLE_CER				
Primal. obj: 1.0000000000e+02 nrm: 1e+00 Viol. con: 0e+00 var: 0e+00				

In the report we see that the variables y1, y3, y4, y5 and two constraints contribute to infeasibility with non-zero values in the Activity column. Therefore

$$(y_1, \dots, y_7) = (-1, 0, -1, 1, 1, 0, 0)$$

is the dual infeasibility certificate as in [Sec. 8.2.1](#). This just means, that along the ray

$$(0, 0, 0, 0, 0, 0, 0) + t(y_1, \dots, y_7) = (-t, 0, -t, t, t, 0, 0), \quad t > 0,$$

which belongs to the feasible set, the objective value $100t$ can be arbitrarily large, i.e. the problem is unbounded.

In the example problem we could

- Add a lower bound on y3. This will directly invalidate the certificate of dual infeasibility.
- Increase the objective coefficient of y3. Changing the coefficients sufficiently will invalidate the inequality $c^T y^* > 0$ and thus the certificate.

7.3.4 Suggestions

Primal infeasibility

When trying to understand what causes the unexpected primal infeasible status use the following hints:

- Remove the objective function. This does not change the infeasibility status but simplifies the problem, eliminating any possibility of issues related to the objective function.
- Remove cones, semidefinite variables and integer constraints. Solve only the linear part of the problem. Typical simple modeling errors will lead to infeasibility already at this stage.
- Consider whether your problem has some obvious 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.
- See if there are any obvious contradictions, for instance a variable is bounded both in the variables and constraints section, and the bounds are contradictory.
- Consider replacing suspicious equality constraints by inequalities. For instance, instead of $x_{12} = 200$ see what happens for $x_{12} \geq 200$ or $x_{12} \leq 200$.

- Relax bounds of the suspicious constraints or variables.
- For integer problems, remove integrality constraints on some/all variables and see if the problem solves.
- Form an **elastic model**: allow to violate constraints at a cost. Introduce slack variables and add them to the objective as penalty. For instance, suppose we have a constraint

$$\begin{array}{ll}\text{minimize} & c^T x, \\ \text{subject to} & a^T x \leq b.\end{array}$$

which might be causing infeasibility. Then create a new variable y and form the problem which contains:

$$\begin{array}{ll}\text{minimize} & c^T x + y, \\ \text{subject to} & a^T x \leq b + y.\end{array}$$

Solving this problem will reveal by how much the constraint needs to be relaxed in order to become feasible. This is equivalent to inspecting the infeasibility certificate but may be more intuitive.

- If you think you have a feasible solution or its part, fix all or some of the variables to those values. Presolve will propagate them through the model and potentially reveal more localized sources of infeasibility.
- Dump the problem in PTF or LP format and verify that the problem that was passed to the optimizer corresponds to the problem expressed in the high-level modeling language, if any such was used.

Dual infeasibility

When trying to understand what causes the unexpected dual infeasible status use the following hints:

- Verify that the objective coefficients are reasonably sized.
- Check if no bounds and constraints are missing, for example if all variables that should be nonnegative have been declared as such etc.
- Strengthen bounds of the suspicious constraints or variables.
- Form an series of models with decreasing bounds on the objective, that is, instead of objective

$$\text{minimize } c^T x$$

solve the problem with an additional constraint such as

$$c^T x = -10^5$$

and inspect the solution to figure out the mechanism behind arbitrarily decreasing objective values. This is equivalent to inspecting the infeasibility certificate but may be more intuitive.

- Dump the problem in PTF or LP format and verify that the problem that was passed to the optimizer corresponds to the problem expressed in the high-level modeling language, if any such was used.

Please note that modifying the problem to invalidate the reported certificate does *not* imply that the problem becomes feasible — the reason for infeasibility may simply *move*, resulting a problem that is still infeasible, but for a different reason. More often, the reported certificate can be used to give a hint about errors or inconsistencies in the model that produced the problem.

7.4 Python Console

The **MOSEK** Python Console is an alternative to the **MOSEK** Command Line Tool. It can be used for interactive loading, solving and debugging optimization problems stored in files, for example **MOSEK** task files. It facilitates debugging techniques described in [Sec. 7](#).

7.4.1 Usage

The tool requires Python 3. The **MOSEK** interface for Python must be installed following the installation instructions for Python API or Python Fusion API. The easiest option is

```
pip install Mosek
```

The Python Console is contained in the file `mosekconsole.py` in the folder with **MOSEK** binaries. It can be copied to an arbitrary location. The file is also available for [download here](#) (`mosekconsole.py`).

To run the console in interactive mode use

```
python mosekconsole.py
```

To run the console in batch mode provide a semicolon-separated list of commands as the second argument of the script, for example:

```
python mosekconsole.py "read data.task.gz; solve form=dual; writesol data"
```

The script is written using the **MOSEK** Python API and can be extended by the user if more specific functionality is required. We refer to the documentation of the Python API.

7.4.2 Examples

To read a problem from `data.task.gz`, solve it, and write solutions to `data.sol`, `data.bas` or `data.itg`:

```
read data.task.gz; solve; writesol data
```

To convert between file formats:

```
read data.task.gz; write data.mps
```

To set a parameter before solving:

```
read data.task.gz; param INTPNT_CO_TOL_DFEAS 1e-9; solve"
```

To list parameter values related to the mixed-integer optimizer in the task file:

```
read data.task.gz; param MIO
```

To print a summary of problem structure:

```
read data.task.gz; anapro
```

To solve a problem forcing the dual and switching off presolve:

```
read data.task.gz; solve form=dual presolve=no
```

To write an infeasible subproblem to a file for debugging purposes:

```
read data.task.gz; solve; infsub; write inf.opf
```

7.4.3 Full list of commands

Below is a brief description of all the available commands. Detailed information about a specific command `cmd` and its options can be obtained with

```
help cmd
```

Table 7.1: List of commands of the MOSEK Python Console.

Command	Description
<code>help [command]</code>	Print list of commands or info about a specific command
<code>log filename</code>	Save the session to a file
<code>intro</code>	Print MOSEK splashscreen
<code>testlic</code>	Test the license system
<code>read filename</code>	Load problem from file
<code>reread</code>	Reload last problem file
<code>solve [options]</code>	Solve current problem
<code>write filename</code>	Write current problem to file
<code>param [name [value]]</code>	Set a parameter or get parameter values
<code>paramdef</code>	Set all parameters to default values
<code>paramdiff</code>	Show parameters with non-default values
<code>info [name]</code>	Get an information item
<code>anapro</code>	Analyze problem data
<code>hist</code>	Plot a histogram of problem data
<code>histsol</code>	Plot a histogram of the solutions
<code>spy</code>	Plot the sparsity pattern of the data matrices
<code>truncate epsilon</code>	Truncate small coefficients down to 0
<code>resobj [fac]</code>	Rescale objective by a factor
<code>anasol</code>	Analyze solutions
<code>removeitg</code>	Remove integrality constraints
<code>removecones</code>	Remove all cones and leave just the linear part
<code>infsub</code>	Replace current problem with its infeasible subproblem
<code>writesol basename</code>	Write solution(s) to file(s) with given basename
<code>del_sol</code>	Remove all solutions from the task
<code>optserver [url]</code>	Use an OptServer to optimize
<code>exit</code>	Leave

Chapter 8

Problem Formulation and Solutions

In this chapter we will discuss the following issues:

- The formal, mathematical formulations of the problem types that **MOSEK** can solve and their duals.
- The solution information produced by **MOSEK**.
- The infeasibility certificate produced by **MOSEK** if the problem is infeasible.

For the underlying mathematical concepts, derivations and proofs see the [Modeling Cookbook](#) or any book on convex optimization. This chapter explains how the related data is organized specifically within the **MOSEK** API. Below is an outline of the various formats. For details see the corresponding subsections.

8.1 Continuous problem formulations

- [Sec. 8.2.1](#)

A linear problem has the form

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

- [Sec. 8.2.2](#)

Conic optimization extends linear optimization with *affine conic constraints* (ACC), so a conic problem has the form

$$\begin{array}{llllll} \text{minimize} & & c^T x + c^f & & & \\ \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \\ & & & Fx + g & \in & \mathcal{D}, \end{array}$$

where \mathcal{D} is a product of domains from [Sec. 11.5](#).

- [Sec. 8.2.3](#)

A conic optimization problem can be further extended with *semidefinite variables*, leading to a semidefinite optimization problem of the form

$$\begin{array}{llllll} \text{minimize} & & c^T x + \langle \overline{C}, \overline{X} \rangle + c^f & & & \\ \text{subject to} & l^c & \leq & Ax + \langle \overline{A}, \overline{X} \rangle & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \\ & & & Fx + \langle \overline{F}, \overline{X} \rangle + g & \in & \mathcal{D}, \\ & & & \overline{X} & \in & \mathcal{S}_+, \end{array}$$

where \mathcal{D} is a product of domains from [Sec. 11.5](#) and \mathcal{S}_+ is a product of PSD cones meaning that \overline{X} is a sequence of PSD matrix variables.

- Sec. 8.2.4

A quadratic problem or quadratically constrained problem has the form

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T Q^o x + c^T x + c^f \\ \text{subject to} & \begin{array}{ll} l^c \leq & \frac{1}{2}x^T Q^c x + Ax \leq u^c, \\ l^x \leq & x \leq u^x. \end{array} \end{array}$$

8.2 Mixed-integer problem formulations

- **Integer variables.** A linear, conic or quadratic problem without semidefinite variables or domains can be extended with the specification of integer variables, that is

$$x_I \in \mathbb{Z}$$

for some index set I . A problem with at least one integer variable is solved by the mixed-integer optimizer.

8.2.1 Linear Optimization

MOSEK accepts linear optimization problems of the form

$$\begin{array}{ll} \text{minimize} & c^T x + c^f \\ \text{subject to} & \begin{array}{ll} l^c \leq & Ax \leq u^c, \\ l^x \leq & x \leq u^x, \end{array} \end{array} \quad (8.1)$$

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.
- $c^f \in \mathbb{R}$ is a constant term in the objective
- $A \in \mathbb{R}^{m \times n}$ 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.

Lower and upper bounds can be infinite, or in other words the corresponding bound may be omitted.

A primal solution (x) is *(primal) feasible* if it satisfies all constraints in (8.1). If (8.1) has at least one primal feasible solution, then (8.1) is said to be (primal) feasible. In case (8.1) does not have a feasible solution, the problem is said to be *(primal) infeasible*.

Duality for Linear Optimization

Corresponding to the primal problem (8.1), there is a dual problem

$$\begin{array}{ll} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f \\ & A^T y + s_l^x - s_u^x = c, \\ \text{subject to} & -y + s_l^c - s_u^c = 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0, \end{array} \quad (8.2)$$

where

- s_l^c are the dual variables for lower bounds of constraints,
- s_u^c are the dual variables for upper bounds of constraints,
- s_l^x are the dual variables for lower bounds of variables,
- s_u^x are the dual variables for upper bounds of variables.

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. This is equivalent to removing the corresponding dual variable from the dual problem. For example:

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

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 (8.2). If (8.2) has at least one feasible solution, then (8.2) is *(dual) feasible*, otherwise the problem is *(dual) infeasible*.

A solution

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

is denoted a *primal-dual feasible solution*, if (x^*) is a solution to the primal problem (8.1) and $(y^*, (s_l^c)^*, (s_u^c)^*, (s_l^x)^*, (s_u^x)^*)$ is a solution to the corresponding dual problem (8.2). We also define an auxiliary vector

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

containing the activities of linear constraints.

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

$$\begin{aligned} & c^T x^* + c^f - \{ (l^c)^T (s_l^c)^* - (u^c)^T (s_u^c)^* + (l^x)^T (s_l^x)^* - (u^x)^T (s_u^x)^* + c^f \} \\ &= \sum_{i=0}^{m-1} [(s_l^c)_i^* ((x_i^c)^* - l_i^c) + (s_u^c)_i^* (u_i^c - (x_i^c)^*)] \\ &+ \sum_{j=0}^{n-1} [(s_l^x)_j^* (x_j^x - l_j^x) + (s_u^x)_j^* (u_j^x - x_j^x)] \geq 0 \end{aligned} \quad (8.3)$$

where the first relation can be obtained by transposing and multiplying the dual constraints (8.2) 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.

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

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

are satisfied.

If (8.1) 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.

Infeasibility for Linear Optimization

Primal Infeasible Problems

If the problem (8.1) is infeasible (has no feasible solution), **MOSEK** will report a certificate of primal infeasibility: The dual solution reported is the certificate of infeasibility, and the primal solution is undefined.

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

$$\begin{aligned} & \text{maximize} && (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x \\ & \text{subject to} && A^T y + s_l^x - s_u^x = 0, \\ & && -y + s_l^c - s_u^c = 0, \\ & && s_l^c, s_u^c, s_l^x, s_u^x \geq 0, \end{aligned} \tag{8.4}$$

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

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

to (8.4) so that

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

Such a solution implies that (8.4) is unbounded, and that (8.1) is infeasible.

Dual Infeasible Problems

If the problem (8.2) is infeasible (has no feasible solution), **MOSEK** will report a certificate of dual infeasibility: The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && \hat{l}^c \leq Ax \leq \hat{u}^c, \\ & && \hat{l}^x \leq x \leq \hat{u}^x, \end{aligned} \tag{8.5}$$

where

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

and

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

such that

$$c^T x < 0.$$

Such a solution implies that (8.5) is unbounded, and that (8.2) is infeasible.

In case that both the primal problem (8.1) and the dual problem (8.2) are infeasible, **MOSEK** will report only one of the two possible certificates — which one is not defined (**MOSEK** returns the first certificate found).

Minimalization vs. Maximalization

When the objective sense of problem (8.1) is maximization, i.e.

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

the objective sense of the dual problem changes to minimization, and the domain of all dual variables changes sign in comparison to (8.2). The dual problem thus takes the form

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

This means that the duality gap, defined in (8.3) as the primal minus the dual objective value, becomes nonpositive. It follows that the dual objective will always be greater than or equal to the primal objective. The primal infeasibility certificate will be reported by **MOSEK** as a solution to the system

$$\begin{aligned} A^T y + s_l^x - s_u^x &= 0, \\ -y + s_l^c - s_u^c &= 0, \\ s_l^c, s_u^c, s_l^x, s_u^x &\leq 0, \end{aligned} \tag{8.6}$$

such that the objective value is strictly negative

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

Similarly, the certificate of dual infeasibility is an x satisfying the requirements of (8.5) such that $c^T x > 0$.

8.2.2 Conic Optimization

Conic optimization is an extension of linear optimization (see [Sec. 8.2.1](#)) allowing conic domains to be specified for affine expressions. A conic optimization problem to be solved by **MOSEK** can be written as

$$\begin{aligned} & \text{minimize} && c^T x + c^f \\ & \text{subject to} && \begin{array}{llll} l^c & \leq & Ax & \leq & u^c, \\ l^x & \leq & x & \leq & u^x, \\ & & Fx + g & \in & \mathcal{D}, \end{array} \end{aligned} \tag{8.7}$$

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.
- $c^f \in \mathbb{R}$ is a constant term in the objective
- $A \in \mathbb{R}^{m \times n}$ 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.

is the same as in [Sec. 8.2.1](#) and moreover:

- $F \in \mathbb{R}^{k \times n}$ is the affine conic constraint matrix.,
- $g \in \mathbb{R}^k$ is the affine conic constraint constant term vector.,
- \mathcal{D} is a Cartesian product of conic domains, namely $\mathcal{D} = \mathcal{D}_1 \times \dots \times \mathcal{D}_p$, where p is the number of individual affine conic constraints (ACCs), and each domain is one from [Sec. 11.5](#).

The total dimension of the domain \mathcal{D} must be equal to k , the number of rows in F and g . Lower and upper bounds can be infinite, or in other words the corresponding bound may be omitted.

MOSEK supports also the cone of positive semidefinite matrices. In order not to obscure this section with additional notation, that extension is discussed in [Sec. 8.2.3](#).

Duality for Conic Optimization

Corresponding to the primal problem (8.7), there is a dual problem

$$\begin{aligned}
& \text{maximize} && (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x - g^T \dot{y} + c^f \\
& \text{subject to} && A^T y + s_l^x - s_u^x + F^T \dot{y} = c, \\
& && -y + s_l^c - s_u^c = 0, \\
& && s_l^c, s_u^c, s_l^x, s_u^x \geq 0, \\
& && \dot{y} \in \mathcal{D}^*,
\end{aligned} \tag{8.8}$$

where

- s_l^c are the dual variables for lower bounds of constraints,
- s_u^c are the dual variables for upper bounds of constraints,
- s_l^x are the dual variables for lower bounds of variables,
- s_u^x are the dual variables for upper bounds of variables,
- \dot{y} are the dual variables for affine conic constraints,
- the dual domain $\mathcal{D}^* = \mathcal{D}_1^* \times \cdots \times \mathcal{D}_p^*$ is a Cartesian product of cones dual to \mathcal{D}_i .

One can check that the dual problem of the dual problem is identical to the original primal problem.

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

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

A solution

$$(y, s_l^c, s_u^c, s_l^x, s_u^x, \dot{y})$$

to the dual problem is feasible if it satisfies all the constraints in (8.8). If (8.8) has at least one feasible solution, then (8.8) is *(dual) feasible*, otherwise the problem is *(dual) infeasible*.

A solution

$$(x^*, y^*, (s_l^c)^*, (s_u^c)^*, (s_l^x)^*, (s_u^x)^*, (\dot{y})^*)$$

is denoted a *primal-dual feasible solution*, if (x^*) is a solution to the primal problem (8.7) and $(y^*, (s_l^c)^*, (s_u^c)^*, (s_l^x)^*, (s_u^x)^*, (\dot{y})^*)$ is a solution to the corresponding dual problem (8.8). We also define an auxiliary vector

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

containing the activities of linear constraints.

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

$$\begin{aligned}
& c^T x^* + c^f - \{ (l^c)^T (s_l^c)^* - (u^c)^T (s_u^c)^* + (l^x)^T (s_l^x)^* - (u^x)^T (s_u^x)^* - g^T (\dot{y})^* + c^f \} \\
& = \sum_{i=0}^{m-1} [(s_l^c)_i^* ((x_i^c)^* - l_i^c) + (s_u^c)_i^* (u_i^c - (x_i^c)^*)] \\
& + \sum_{j=0}^{n-1} [(s_l^x)_j^* (x_j - l_j^x) + (s_u^x)_j^* (u_j^x - x_j^*)] \\
& + ((\dot{y})^*)^T (F x^* + g) \geq 0
\end{aligned} \tag{8.9}$$

where the first relation can be obtained by transposing and multiplying the dual constraints (8.2) 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.

It is well-known that, under some non-degeneracy assumptions that exclude ill-posed cases, a conic optimization problem has an optimal solution if and only if there exist feasible primal-dual solution so that the duality gap is zero, or, equivalently, that the *complementarity conditions*

$$\begin{aligned} (s_l^c)_i^* ((x_i^c)^* - l_i^c) &= 0, & i = 0, \dots, m-1, \\ (s_u^c)_i^* (u_i^c - (x_i^c)^*) &= 0, & i = 0, \dots, m-1, \\ (s_l^x)_j^* (x_j^* - l_j^x) &= 0, & j = 0, \dots, n-1, \\ (s_u^x)_j^* (u_j^x - x_j^*) &= 0, & j = 0, \dots, n-1, \\ ((y)^*)^T (Fx^* + g) &= 0, \end{aligned} \tag{8.10}$$

are satisfied.

If (8.7) 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.

Infeasibility for Conic Optimization

Primal Infeasible Problems

If the problem (8.7) is infeasible (has no feasible solution), **MOSEK** will report a certificate of primal infeasibility: The dual solution reported is the certificate of infeasibility, and the primal solution is undefined.

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

$$\begin{aligned} &\text{maximize} && (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x - g^T \dot{y} \\ &\text{subject to} && A^T y + s_l^x - s_u^x + F^T \dot{y} = 0, \\ & && -y + s_l^c - s_u^c = 0, \\ & && s_l^c, s_u^c, s_l^x, s_u^x \geq 0, \\ & && \dot{y} \in \mathcal{D}^*, \end{aligned} \tag{8.11}$$

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

$$(y^*, (s_l^c)^*, (s_u^c)^*, (s_l^x)^*, (s_u^x)^*, (\dot{y})^*)$$

to (8.11) so that

$$(l^c)^T (s_l^c)^* - (u^c)^T (s_u^c)^* + (l^x)^T (s_l^x)^* - (u^x)^T (s_u^x)^* - g^T \dot{y}^* > 0.$$

Such a solution implies that (8.11) is unbounded, and that (8.7) is infeasible.

Dual Infeasible Problems

If the problem (8.8) is infeasible (has no feasible solution), **MOSEK** will report a certificate of dual infeasibility: The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

$$\begin{aligned} &\text{minimize} && c^T x \\ &\text{subject to} && \hat{l}^c \leq Ax \leq \hat{u}^c, \\ & && \hat{l}^x \leq x \leq \hat{u}^x, \\ & && Fx \in \mathcal{D} \end{aligned} \tag{8.12}$$

where

$$\hat{l}_i^c = \begin{cases} 0 & \text{if } l_i^c > -\infty, \\ -\infty & \text{otherwise,} \end{cases} \quad \text{and} \quad \hat{u}_i^c := \begin{cases} 0 & \text{if } u_i^c < \infty, \\ \infty & \text{otherwise,} \end{cases} \tag{8.13}$$

and

$$\hat{l}_j^x = \begin{cases} 0 & \text{if } l_j^x > -\infty, \\ -\infty & \text{otherwise,} \end{cases} \quad \text{and} \quad \hat{u}_j^x := \begin{cases} 0 & \text{if } u_j^x < \infty, \\ \infty & \text{otherwise,} \end{cases} \tag{8.14}$$

such that

$$c^T x < 0.$$

Such a solution implies that (8.12) is unbounded, and that (8.8) is infeasible.

In case that both the primal problem (8.7) and the dual problem (8.8) are infeasible, **MOSEK** will report only one of the two possible certificates — which one is not defined (**MOSEK** returns the first certificate found).

Minimalization vs. Maximalization

When the objective sense of problem (8.7) is maximization, i.e.

$$\begin{array}{llll} \text{maximize} & & c^T x + c^f \\ \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \\ & & & Fx + g & \in & \mathcal{D}, \end{array}$$

the objective sense of the dual problem changes to minimization, and the domain of all dual variables changes sign in comparison to (8.2). The dual problem thus takes the form

$$\begin{array}{ll} \text{minimize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x - g^T \dot{y} + c^f \\ \text{subject to} & A^T y + s_l^c - s_u^c + F^T \dot{y} = c, \\ & -y + s_l^c - s_u^c = 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \leq 0, \\ & -\dot{y} \in \mathcal{D}^* \end{array}$$

This means that the duality gap, defined in (8.9) as the primal minus the dual objective value, becomes nonpositive. It follows that the dual objective will always be greater than or equal to the primal objective. The primal infeasibility certificate will be reported by **MOSEK** as a solution to the system

$$\begin{array}{l} A^T y + s_l^c - s_u^c + F^T \dot{y} = 0, \\ -y + s_l^c - s_u^c = 0, \\ s_l^c, s_u^c, s_l^x, s_u^x \leq 0, \\ -\dot{y} \in \mathcal{D}^* \end{array} \quad (8.15)$$

such that the objective value is strictly negative

$$(l^c)^T (s_l^c)^* - (u^c)^T (s_u^c)^* + (l^x)^T (s_l^x)^* - (u^x)^T (s_u^x)^* - g^T \dot{y} < 0.$$

Similarly, the certificate of dual infeasibility is an x satisfying the requirements of (8.12) such that $c^T x > 0$.

8.2.3 Semidefinite Optimization

Semidefinite optimization is an extension of conic optimization (see Sec. 8.2.2) allowing positive semidefinite matrix variables to be used in addition to the usual scalar variables. All the other parts of the input are defined exactly as in Sec. 8.2.2, and the discussion from that section applies verbatim to all properties of problems with semidefinite variables. We only briefly indicate how the corresponding formulae should be modified with semidefinite terms.

A semidefinite optimization problem can be written as

$$\begin{array}{llll} \text{minimize} & & c^T x + \langle \overline{C}, \overline{X} \rangle + c^f \\ \text{subject to} & l^c & \leq & Ax + \langle \overline{A}, \overline{X} \rangle & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \\ & & & Fx + \langle \overline{F}, \overline{X} \rangle + g & \in & \mathcal{D}, \\ & & & \overline{X}_j & \in & \mathcal{S}_+^j, j = 1, \dots, s \end{array}$$

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.
- $c^f \in \mathbb{R}$ is a constant term in the objective
- $A \in \mathbb{R}^{m \times n}$ 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.
- $F \in \mathbb{R}^{k \times n}$ is the affine conic constraint matrix.,
- $g \in \mathbb{R}^k$ is the affine conic constraint constant term vector.,
- \mathcal{D} is a Cartesian product of conic domains, namely $\mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_p$, where p is the number of individual affine conic constraints (ACCs), and each domain is one from [Sec. 11.5](#).

is the same as in [Sec. 8.2.2](#) and moreover:

- there are s symmetric positive semidefinite variables, the j -th of which is $\bar{X}_j \in \mathcal{S}_+^{r_j}$ of dimension r_j ,
- $\bar{C} = (\bar{C}_j)_{j=1,\dots,s}$ is a collection of symmetric coefficient matrices in the objective, with $\bar{C}_j \in \mathcal{S}^{r_j}$, and we interpret the notation $\langle \bar{C}, \bar{X} \rangle$ as a shorthand for

$$\langle \bar{C}, \bar{X} \rangle := \sum_{j=1}^s \langle \bar{C}_j, \bar{X}_j \rangle.$$

- $\bar{A} = (\bar{A}_{ij})_{i=1,\dots,m,j=1,\dots,s}$ is a collection of symmetric coefficient matrices in the constraints, with $\bar{A}_{ij} \in \mathcal{S}^{r_j}$, and we interpret the notation $\langle \bar{A}, \bar{X} \rangle$ as a shorthand for the vector

$$\langle \bar{A}, \bar{X} \rangle := \left(\sum_{j=1}^s \langle \bar{A}_{ij}, \bar{X}_j \rangle \right)_{i=1,\dots,m}.$$

- $\bar{F} = (\bar{F}_{ij})_{i=1,\dots,k,j=1,\dots,s}$ is a collection of symmetric coefficient matrices in the affine conic constraints, with $\bar{F}_{ij} \in \mathcal{S}^{r_j}$, and we interpret the notation $\langle \bar{F}, \bar{X} \rangle$ as a shorthand for the vector

$$\langle \bar{F}, \bar{X} \rangle := \left(\sum_{j=1}^s \langle \bar{F}_{ij}, \bar{X}_j \rangle \right)_{i=1,\dots,k}.$$

In each case the matrix inner product between symmetric matrices of the same dimension r is defined as

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

To summarize, above the formulation extends that from [Sec. 8.2.2](#) by the possibility of including semidefinite terms in the objective, constraints and affine conic constraints.

Duality

The definition of the dual problem (8.8) becomes:

$$\begin{aligned}
& \text{maximize} && (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x - g^T \dot{y} + c^f \\
& \text{subject to} && \\
& && A^T y + s_l^x - s_u^x + F^T \dot{y} = c, \\
& && -y + s_l^c - s_u^c = 0, \\
& && \bar{C}_j - \sum_{i=1}^m y_i \bar{A}_{ij} - \sum_{i=1}^k \dot{y}_i \bar{F}_{ij} = S_j, \quad j = 1, \dots, s, \\
& && s_l^c, s_u^c, s_l^x, s_u^x \geq 0, \\
& && \dot{y} \in \mathcal{D}^*, \\
& && \bar{S}_j \in \mathcal{S}_+^{r_j}, \quad j = 1, \dots, s.
\end{aligned} \tag{8.16}$$

Complementarity conditions (8.10) include the additional relation:

$$\langle \bar{X}_j, \bar{S}_j \rangle = 0 \quad j = 1, \dots, s. \tag{8.17}$$

Infeasibility

A certificate of primal infeasibility (8.11) is now a feasible solution to:

$$\begin{aligned}
& \text{maximize} && (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x - g^T \dot{y} \\
& \text{subject to} && \\
& && A^T y + s_l^x - s_u^x + F^T \dot{y} = 0, \\
& && -y + s_l^c - s_u^c = 0, \\
& && -\sum_{i=1}^m y_i \bar{A}_{ij} - \sum_{i=1}^k \dot{y}_i \bar{F}_{ij} = S_j, \quad j = 1, \dots, s, \\
& && s_l^c, s_u^c, s_l^x, s_u^x \geq 0, \\
& && \dot{y} \in \mathcal{D}^*, \\
& && \bar{S}_j \in \mathcal{S}_+^{r_j}, \quad j = 1, \dots, s.
\end{aligned} \tag{8.18}$$

such that the objective value is strictly positive.

Similarly, a dual infeasibility certificate (8.12) is a feasible solution to

$$\begin{aligned}
& \text{minimize} && c^T x + \langle \bar{C}, \bar{X} \rangle \\
& \text{subject to} && \hat{l}^c \leq Ax + \langle \bar{A}, \bar{X} \rangle \leq \hat{u}^c, \\
& && \hat{l}^x \leq x \leq \hat{u}^x, \\
& && Fx + \langle \bar{F}, \bar{X} \rangle \in \mathcal{D}, \\
& && \bar{X}_j \in \mathcal{S}_+^{r_j}, j = 1, \dots, s
\end{aligned} \tag{8.19}$$

where the modified bounds are as in (8.13) and (8.14) and the objective value is strictly negative.

8.2.4 Quadratic and Quadratically Constrained Optimization

A convex quadratic and quadratically constrained optimization problem has the form

$$\begin{aligned}
& \text{minimize} && \frac{1}{2} x^T Q^o x + c^T x + c^f \\
& \text{subject to} && \begin{aligned} l_k^c &\leq \frac{1}{2} x^T Q^k x + \sum_{j=0}^{n-1} a_{kj} x_j \leq u_k^c, & k = 0, \dots, m-1, \\ l_j^x &\leq x_j \leq u_j^x, & j = 0, \dots, n-1, \end{aligned}
\end{aligned} \tag{8.20}$$

where all variables and bounds have the same meaning as for linear problems (see Sec. 8.2.1) and Q^o and all Q^k are symmetric matrices. Moreover, for convexity, Q^o must be a positive semidefinite matrix and Q^k must satisfy

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

The convexity requirement is very important and **MOSEK** checks whether it is fulfilled.

A Recommendation

Any convex quadratic optimization problem can be reformulated as a conic quadratic optimization problem, see [Modeling Cookbook](#) and [And13]. In fact **MOSEK** does such conversion internally as a part of the solution process for the following reasons:

- the conic optimizer is numerically more robust than the one for quadratic problems.
- the conic optimizer is usually faster because quadratic cones are simpler than quadratic functions, even though the conic reformulation usually has more constraints and variables than the original quadratic formulation.
- it is easy to dualize the conic formulation if deemed worthwhile potentially leading to (huge) computational savings.

However, instead of relying on the automatic reformulation we recommend to formulate the problem as a conic problem from scratch because:

- it saves the computational overhead of the reformulation including the convexity check. A conic problem is convex by construction and hence no convexity check is needed for conic problems.
- usually the modeler can do a better reformulation than the automatic method because the modeler can exploit the knowledge of the problem at hand.

To summarize we recommend to formulate quadratic problems and in particular quadratically constrained problems directly in conic form.

Duality for Quadratic and Quadratically Constrained Optimization

The dual problem corresponding to the quadratic and quadratically constrained optimization problem (8.20) is given by

$$\begin{aligned}
 & \text{maximize} && (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + \frac{1}{2} x^T \left\{ \sum_{k=0}^{m-1} y_k Q^k - Q^o \right\} x + c^f \\
 & \text{subject to} && A^T y + s_l^x - s_u^x + \left\{ \sum_{k=0}^{m-1} y_k Q^k - Q^o \right\} x = c, \\
 & && -y + s_l^c - s_u^c = 0, \\
 & && s_l^c, s_u^c, s_l^x, s_u^x \geq 0.
 \end{aligned} \tag{8.21}$$

The dual problem is related to the dual problem for linear optimization (see [Sec. 8.2.1](#)), but depends on the variable x which in general can not be eliminated. In the solutions reported by **MOSEK**, the value of x is the same for the primal problem (8.20) and the dual problem (8.21).

Infeasibility for Quadratic Optimization

In case **MOSEK** finds a problem to be infeasible it reports a certificate of infeasibility. We write them out explicitly for quadratic problems, that is when $Q^k = 0$ for all k and quadratic terms appear only in the objective Q^o . In this case the constraints both in the primal and dual problem are linear, and **MOSEK** produces for them the same infeasibility certificate as for linear problems.

The certificate of primal infeasibility is a solution to the problem (8.4) such that the objective value is strictly positive.

The certificate of dual infeasibility is a solution to the problem (8.5) together with an additional constraint

$$Q^o x = 0$$

such that the objective value is strictly negative.

Chapter 9

Optimizers

The most essential part of **MOSEK** are the optimizers:

- *primal simplex* (linear problems),
- *dual simplex* (linear problems),
- *interior-point* (linear, quadratic and conic problems),
- *mixed-integer* (problems with integer variables).

The structure of a successful optimization process is roughly:

- **Presolve**
 1. *Elimination*: Reduce the size of the problem.
 2. *Dualizer*: Choose whether to solve the primal or the dual form of the problem.
 3. *Scaling*: Scale the problem for better numerical stability.
- **Optimization**
 1. *Optimize*: Solve the problem using selected method.
 2. *Terminate*: Stop the optimization when specific termination criteria have been met.
 3. *Report*: Return the solution or an infeasibility certificate.

The preprocessing stage is transparent to the user, but useful to know about for tuning purposes. The purpose of the preprocessing steps is to make the actual optimization more efficient and robust. We discuss the details of the above steps in the following sections.

9.1 Presolve

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

1. remove redundant constraints,
2. eliminate fixed variables,
3. remove linear dependencies,
4. substitute out (implied) free variables, and
5. 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 [AA95] and [AGMeszarosX96].

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`.

In the following we describe in more detail the presolve applied to continuous, i.e., linear and conic optimization problems, see Sec. 9.2 and Sec. 9.3. The mixed-integer optimizer, Sec. 9.4, applies similar techniques. The two most time-consuming steps of the presolve for continuous optimization problems are

- the eliminator, and
- the linear dependency check.

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

Numerical issues in the presolve

During the presolve the problem is reformulated so that it hopefully solves faster. However, in rare cases the presolved problem may be harder to solve than the original problem. The presolve may also be infeasible although the original problem is not. If it is suspected that presolved problem is much harder to solve than the original, we suggest to first turn the eliminator off by setting the parameter `MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES` to 0. If that does not help, then trying to turn entire presolve off may help.

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

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 by setting the parameter `MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES` to 0. In rare cases the eliminator may cause that the problem becomes much hard to solve.

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 practice to build models without linear dependencies, but that is not always easy for the user to control. 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`.

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 more efficient to solve the primal or dual problem. The form (primal or dual) is displayed in the **MOSEK** log and available as an information item from the solver. 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 and conic (but not semidefinite) problems may be automatically dualized.

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 data. 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 of this issue is to reformulate the problem, 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.

9.2 Linear Optimization

9.2.1 Optimizer Selection

Two different types of optimizers are available for linear problems: The default is an interior-point method, and the alternative is the simplex method (primal or dual). The optimizer can be selected using the parameter `MSK_IPAR_OPTIMIZER`.

The Interior-point or the Simplex Optimizer?

Given a linear optimization problem, which optimizer is the best: the 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 warm-start. On the other hand the simplex method can take advantage of an initial solution, but is less predictable from cold-start. The interior-point optimizer is used by default.

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, make it faster on average than the primal version. 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 one of the simplex variants automatically.

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

9.2.2 The Interior-point Optimizer

The purpose of this section is to provide information about the algorithm employed in the **MOSEK** interior-point optimizer for linear problems and about its termination criteria.

The homogeneous primal-dual problem

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

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && Ax = b, \\ & && x \geq 0. \end{aligned} \tag{9.1}$$

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

Since it is not known beforehand whether problem (9.1) has an optimal solution, is primal infeasible or is dual infeasible, the optimization algorithm must deal with all three situations. This is the reason why **MOSEK** solves the so-called homogeneous model

$$\begin{aligned} Ax - b\tau &= 0, \\ A^T y + s - c\tau &= 0, \\ -c^T x + b^T y - \kappa &= 0, \\ x, s, \tau, \kappa &\geq 0, \end{aligned} \tag{9.2}$$

where y and s correspond to the dual variables in (9.1), and τ and κ are two additional scalar variables. Note that the homogeneous model (9.2) 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 (9.2) satisfies

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

Moreover, there is always a solution that has the property $\tau^* + \kappa^* > 0$.

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

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

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

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

is a primal-dual optimal solution (see [Sec. 8.2.1](#) for the mathematical background on duality and optimality).

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

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

This implies that at least one of

$$c^T x^* < 0 \quad (9.3)$$

or

$$b^T y^* > 0 \quad (9.4)$$

is satisfied. If (9.3) is satisfied then x^* is a certificate of dual infeasibility, whereas if (9.4) is satisfied then y^* is a certificate of primal 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 [And09].

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 the k -th iteration of the interior-point algorithm a trial solution

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

to homogeneous model is generated, where

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

Optimal case

Whenever the trial solution satisfies the criterion

$$\begin{aligned} \left\| A \frac{x^k}{\tau^k} - b \right\|_\infty &\leq \epsilon_p (1 + \|b\|_\infty), \\ \left\| A^T \frac{y^k}{\tau^k} + \frac{s^k}{\tau^k} - c \right\|_\infty &\leq \epsilon_d (1 + \|c\|_\infty), \text{ and} \\ \min \left(\frac{(x^k)^T s^k}{(\tau^k)^2}, \left| \frac{c^T x^k}{\tau^k} - \frac{b^T y^k}{\tau^k} \right| \right) &\leq \epsilon_g \max \left(1, \frac{\min(|c^T x^k|, |b^T y^k|)}{\tau^k} \right), \end{aligned} \quad (9.5)$$

the interior-point optimizer is terminated and

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

is reported as the primal-dual optimal solution. The interpretation of (9.5) is that the optimizer is terminated if

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

Dual infeasibility certificate

On the other hand, if the trial solution satisfies

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

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

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

and define

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

It is easy to verify that

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

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

Primal infeasibility certificate

Finally, if

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

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

Adjusting optimality criteria

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

Table 9.1: Parameters employed in termination criterion

ToleranceParameter	name
ε_p	<i>MSK_DPAR_INTPNT_TOL_PFEAS</i>
ε_d	<i>MSK_DPAR_INTPNT_TOL_DFEAS</i>
ε_g	<i>MSK_DPAR_INTPNT_TOL_REL_GAP</i>
ε_i	<i>MSK_DPAR_INTPNT_TOL_INFEAS</i>

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 (9.5) reveals that the quality of the solution depends on $\|b\|_\infty$ and $\|c\|_\infty$; the smaller the norms are, the better the solution accuracy.

The interior-point method as implemented by **MOSEK** will converge toward optimality and primal and dual feasibility at the same rate [And09]. 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 , ε_g and ε_i , have to be relaxed together to achieve an effect.

The basis identification discussed in Sec. 9.2.2 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.

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 [AY96]. In the following we provide an overall idea of the procedure.

There are some cases in which a basic solution could be more valuable:

- a basic solution is often more accurate than an interior-point solution,
- a basic solution can be used to warm-start the simplex algorithm in case of reoptimization,
- a basic solution is in general more sparse, i.e. more variables are fixed to zero. This is particularly appealing when solving continuous relaxations of mixed integer problems, as well as in all applications in which sparser solutions are preferred.

To illustrate how the basis identification routine works, we use the following trivial example:

$$\begin{array}{ll} \text{minimize} & x + y \\ \text{subject to} & x + y = 1, \\ & x, y \geq 0. \end{array}$$

It is easy to see that all feasible solutions are also optimal. In particular, there are two basic solutions, namely

$$\begin{aligned}(x_1^*, y_1^*) &= (1, 0), \\ (x_2^*, y_2^*) &= (0, 1).\end{aligned}$$

The interior point algorithm will actually converge to the center of the optimal set, i.e. to $(x^*, y^*) = (1/2, 1/2)$ (to see this in **MOSEK** deactivate *Presolve*).

In practice, when the algorithm gets close to the optimal solution, it is possible to construct in polynomial time an initial basis for the simplex algorithm from the current interior point solution. This basis is used to warm-start the simplex algorithm that will provide the optimal basic solution. In most cases the constructed basis is optimal, or very few iterations are required by the simplex algorithm to make it optimal and hence the final *clean-up* phase be short. However, for some cases of ill-conditioned problems the additional simplex clean up phase may take of lot a time.

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*

control when basis identification is performed.

The type of simplex algorithm to be used (primal/dual) can be tuned with the parameter `MSK_IPAR_BI_CLEAN_OPTIMIZER`, and the maximum number of iterations can be set with `MSK_IPAR_BI_MAX_ITERATIONS`.

Finally, it should be mentioned that there is no guarantee on which basic solution will be returned.

The Interior-point Log

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

```

Optimizer - threads : 1
Optimizer - solved problem : the dual
Optimizer - Constraints : 2
Optimizer - Cones : 0
Optimizer - Scalar variables : 6 conic : 0
Optimizer - Semi-definite variables : 0 scalarized : 0
Factor - setup time : 0.00 dense det. time : 0.00
Factor - ML order time : 0.00 GP order time : 0.00
Factor - nonzeros before factor : 3 after factor : 3
Factor - dense dim. : 0 flops : 7.
↪00e+001
ITE PFEAS DFEAS GFEAS PRSTATUS POBJ DOBJ MU
↪ TIME
0 1.0e+000 8.6e+000 6.1e+000 1.00e+000 0.000000000e+000 -2.208000000e+003 1.
↪0e+000 0.00
1 1.1e+000 2.5e+000 1.6e-001 0.00e+000 -7.901380925e+003 -7.394611417e+003 2.
↪5e+000 0.00
2 1.4e-001 3.4e-001 2.1e-002 8.36e-001 -8.113031650e+003 -8.055866001e+003 3.3e-
↪001 0.00
3 2.4e-002 5.8e-002 3.6e-003 1.27e+000 -7.777530698e+003 -7.766471080e+003 5.7e-
↪002 0.01
4 1.3e-004 3.2e-004 2.0e-005 1.08e+000 -7.668323435e+003 -7.668207177e+003 3.2e-
↪004 0.01

```

(continues on next page)

(continued from previous page)

```
5    1.3e-008 3.2e-008 2.0e-009 1.00e+000 -7.668000027e+003 -7.668000015e+003 3.2e-
↪008 0.01
6    1.3e-012 3.2e-012 2.0e-013 1.00e+000 -7.667999994e+003 -7.667999994e+003 3.2e-
↪012 0.01
```

The first line displays the number of threads used by the optimizer and the second line indicates if the optimizer chose to solve the primal or dual problem (see [MSK_IPAR_INTPNT_SOLVE_FORM](#)). The next lines display 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 [Sec. 9.2.2](#) the columns of the iteration log have the following meaning:

- **ITE**: Iteration index k .
- **PFEAS**: $\|Ax^k - b\tau^k\|_\infty$. The numbers in this column should converge monotonically towards zero but may stall at low level due to rounding errors.
- **DFEAS**: $\|A^T y^k + s^k - c\tau^k\|_\infty$. The numbers in this column should converge monotonically towards zero but may stall at low level due to rounding errors.
- **GFEAS**: $|-c^T x^k + b^T y^k - \kappa^k|$. The numbers in this column should converge monotonically towards zero but may stall at low level due to rounding errors.
- **PRSTATUS**: This number converges to 1 if the problem has an optimal solution whereas it converges to -1 if that is not the case.
- **POBJ**: $c^T x^k / \tau^k$. An estimate for the primal objective value.
- **DOBJ**: $b^T y^k / \tau^k$. An estimate for the dual objective value.
- **MU**: $\frac{(x^k)^T s^k + \tau^k \kappa^k}{n+1}$. The numbers in this column should always converge to zero.
- **TIME**: Time spent since the optimization started.

9.2.3 The Simplex 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 [Sec. 9.2.1](#) for a discussion. **MOSEK** provides both a primal and a dual variant of the simplex optimizer.

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 [Sec. 8.2.1](#) for a definition of the primal and dual problem. Due to the fact that computations are performed in finite precision **MOSEK** allows violations of primal and dual feasibility within certain tolerances. The user can control the allowed primal and dual tolerances with the parameters [MSK_DPAR_BASIS_TOL_X](#) and [MSK_DPAR_BASIS_TOL_S](#).

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. The same parameter can also be used to force one of the variants.

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 *warm-start*. If the user is solving a sequence of optimization problems by solving the problem, making modifications, and solving again, **MOSEK** will warm-start automatically.

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

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** treats a “numerically 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 a way to escape long sequences where the optimizer tries to recover from an unstable situation.

Examples of set-backs are: 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 it 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: 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 warm-starts, in rare cases switching off this feature may improve stability. This is controlled by the `MSK_IPAR_SIM_HOTSTART` parameter.
- Increase maximum number of 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.

The Simplex Log

Below is a typical log output from the simplex optimizer:

Optimizer	- solved problem	:	the primal			
Optimizer	- Constraints	:	667			
Optimizer	- Scalar variables	:	1424	conic	:	0
Optimizer	- hotstart	:	no			
ITER	DEGITER(%)	PFEAS	DFEAS	POBJ	DOBJ	
↪	TIME	TOTTIME				
0	0.00	1.43e+05	NA	6.5584140832e+03	NA	↪
↪	0.00	0.02				
1000	1.10	0.00e+00	NA	1.4588289726e+04	NA	↪
↪	0.13	0.14				
2000	0.75	0.00e+00	NA	7.3705564855e+03	NA	↪
↪	0.21	0.22				

(continues on next page)

(continued from previous page)

3000	0.67	0.00e+00	NA	6.0509727712e+03	NA	␣
↪	0.29	0.31				
4000	0.52	0.00e+00	NA	5.5771203906e+03	NA	␣
↪	0.38	0.39				
4533	0.49	0.00e+00	NA	5.5018458883e+03	NA	␣
↪	0.42	0.44				

The first lines summarize the problem the optimizer is solving. This is followed by the iteration log, with the following meaning:

- **ITER**: Number of iterations.
- **DEGITER(%)**: Ratio of degenerate iterations.
- **PFEAS**: Primal feasibility measure reported by the simplex optimizer. The numbers should be 0 if the problem is primal feasible (when the primal variant is used).
- **DFEAS**: Dual feasibility measure reported by the simplex optimizer. The number should be 0 if the problem is dual feasible (when the dual variant is used).
- **POBJ**: An estimate for the primal objective value (when the primal variant is used).
- **DOBJ**: An estimate for the dual objective value (when the dual variant is used).
- **TIME**: Time spent since this instance of the simplex optimizer was invoked (in seconds).
- **TOTTIME**: Time spent since optimization started (in seconds).

9.3 Conic Optimization - Interior-point optimizer

For conic optimization problems only an interior-point type optimizer is available.

9.3.1 The homogeneous primal-dual problem

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 [ART03]. In order to keep our discussion simple we will assume that **MOSEK** solves a conic optimization problem of the form:

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && Ax = b, \\ & && x \in \mathcal{K} \end{aligned} \tag{9.6}$$

where \mathcal{K} is a convex cone. The corresponding dual problem is

$$\begin{aligned} & \text{maximize} && b^T y \\ & \text{subject to} && A^T y + s = c, \\ & && s \in \mathcal{K}^* \end{aligned} \tag{9.7}$$

where \mathcal{K}^* is the dual cone of \mathcal{K} . See Sec. 8.2.2 for definitions.

Since it is not known beforehand whether problem (9.6) 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

$$\begin{aligned} Ax - b\tau &= 0, \\ A^T y + s - c\tau &= 0, \\ -c^T x + b^T y - \kappa &= 0, \\ x &\in \mathcal{K}, \\ s &\in \mathcal{K}^*, \\ \tau, \kappa &\geq 0, \end{aligned} \tag{9.8}$$

where y and s correspond to the dual variables in (9.6), and τ and κ are two additional scalar variables. Note that the homogeneous model (9.8) always has a 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 (9.8) satisfies

$$(x^*)^T s^* + \tau^* \kappa^* = 0$$

i.e. complementarity. Observe that $x^* \in \mathcal{K}$ and $s^* \in \mathcal{K}^*$ implies

$$(x^*)^T s^* \geq 0$$

and therefore

$$\tau^* \kappa^* = 0.$$

since $\tau^*, \kappa^* \geq 0$. Hence, at least one of τ^* and κ^* is zero.

First, assume that $\tau^* > 0$ and hence $\kappa^* = 0$. It follows that

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

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

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

is a primal-dual optimal solution.

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

$$\begin{aligned} Ax^* &= 0, \\ A^T y^* + s^* &= 0, \\ -c^T x^* + b^T y^* &= \kappa^*, \\ x^* &\in \mathcal{K}, \\ s^* &\in \mathcal{K}^*. \end{aligned}$$

This implies that at least one of

$$c^T x^* < 0 \tag{9.9}$$

or

$$b^T y^* > 0 \tag{9.10}$$

holds. If (9.9) is satisfied, then x^* is a certificate of dual infeasibility, whereas if (9.10) holds then y^* is a certificate of primal 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 [And09].

9.3.2 Interior-point Termination Criterion

Since computations are performed in finite precision, and for efficiency reasons, it is not possible to solve the homogeneous model exactly in general. 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 the homogeneous model is generated, where

$$x^k \in \mathcal{K}, s^k \in \mathcal{K}^*, \tau^k, \kappa^k > 0.$$

Therefore, it is possible to compute the values:

$$\begin{aligned} \rho_p^k &= \arg \min_{\rho} \left\{ \rho \mid \left\| A \frac{x^k}{\tau^k} - b \right\|_{\infty} \leq \rho \varepsilon_p (1 + \|b\|_{\infty}) \right\}, \\ \rho_d^k &= \arg \min_{\rho} \left\{ \rho \mid \left\| A^T \frac{y^k}{\tau^k} + \frac{s^k}{\tau^k} - c \right\|_{\infty} \leq \rho \varepsilon_d (1 + \|c\|_{\infty}) \right\}, \\ \rho_g^k &= \arg \min_{\rho} \left\{ \rho \mid \left(\frac{(x^k)^T s^k}{(\tau^k)^2}, \left| \frac{c^T x^k}{\tau^k} - \frac{b^T y^k}{\tau^k} \right| \right) \leq \rho \varepsilon_g \max \left(1, \frac{\min(|c^T x^k|, |b^T y^k|)}{\tau^k} \right) \right\}, \\ \rho_{pi}^k &= \arg \min_{\rho} \left\{ \rho \mid \left\| A^T y^k + s^k \right\|_{\infty} \leq \rho \varepsilon_i b^T y^k, b^T y^k > 0 \right\} \text{ and} \\ \rho_{di}^k &= \arg \min_{\rho} \left\{ \rho \mid \left\| Ax^k \right\|_{\infty} \leq -\rho \varepsilon_i c^T x^k, c^T x^k < 0 \right\}. \end{aligned}$$

Note $\varepsilon_p, \varepsilon_d, \varepsilon_g$ and ε_i are nonnegative user specified tolerances.

Optimal Case

Observe ρ_p^k measures how far x^k/τ^k is from being a good approximate primal feasible solution. Indeed if $\rho_p^k \leq 1$, then

$$\left\| A \frac{x^k}{\tau^k} - b \right\|_{\infty} \leq \varepsilon_p (1 + \|b\|_{\infty}). \quad (9.11)$$

This shows the violations in the primal equality constraints for the solution x^k/τ^k is small compared to the size of b given ε_p is small.

Similarly, if $\rho_d^k \leq 1$, then $(y^k, s^k)/\tau^k$ is an approximate dual feasible solution. If in addition $\rho_g \leq 1$, then the solution $(x^k, y^k, s^k)/\tau^k$ is approximate optimal because the associated primal and dual objective values are almost identical.

In other words if $\max(\rho_p^k, \rho_d^k, \rho_g^k) \leq 1$, then

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

is an approximate optimal solution.

Dual Infeasibility Certificate

Next assume that $\rho_{di}^k \leq 1$ and hence

$$\|Ax^k\|_{\infty} \leq -\varepsilon_i c^T x^k \text{ and } -c^T x^k > 0$$

holds. Now in this case 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. Let

$$\bar{x} := \frac{x^k}{-c^T x^k}$$

and it is easy to verify that

$$\|A\bar{x}\|_{\infty} \leq \varepsilon_i \text{ and } c^T \bar{x} = -1$$

which shows \bar{x} is an approximate certificate of dual infeasibility, where ε_i controls the quality of the approximation.

Primal Infeasibility Certificate

Next assume that $\rho_{pi}^k \leq 1$ and hence

$$\|A^T y^k + s^k\|_\infty \leq \varepsilon_i b^T y^k \text{ and } b^T y^k > 0$$

holds. Now in this case the problem is declared primal infeasible and (y^k, s^k) is reported as a certificate of primal infeasibility. The motivation for this stopping criterion is as follows. Let

$$\bar{y} := \frac{y^k}{b^T y^k} \text{ and } \bar{s} := \frac{s^k}{b^T y^k}$$

and it is easy to verify that

$$\|A^T \bar{y} + \bar{s}\|_\infty \leq \varepsilon_i \text{ and } b^T \bar{y} = 1$$

which shows (y^k, s^k) is an approximate certificate of dual infeasibility, where ε_i controls the quality of the approximation.

9.3.3 Adjusting optimality criteria

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

Table 9.2: Parameters employed in termination criterion

ToleranceParameter	name
ε_p	<i>MSK_DPAR_INTPNT_CO_TOL_PFEAS</i>
ε_d	<i>MSK_DPAR_INTPNT_CO_TOL_DFEAS</i>
ε_g	<i>MSK_DPAR_INTPNT_CO_TOL_REL_GAP</i>
ε_i	<i>MSK_DPAR_INTPNT_CO_TOL_INFEAS</i>

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 (9.11) reveals that the quality of the solution depends on $\|b\|_\infty$ and $\|c\|_\infty$; the smaller the norms are, the better the solution accuracy.

The interior-point method as implemented by **MOSEK** will converge toward optimality and primal and dual feasibility at the same rate [And09]. 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 , ε_g and ε_i , have to be relaxed together to achieve an effect.

If the optimizer terminates without locating a solution that satisfies the termination criteria, for example because of a stall or other numerical issues, then it will check if the solution found up to that point satisfies the same criteria with all tolerances multiplied by the value of *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*. If this is the case, the solution is still declared as optimal.

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

9.3.4 The Interior-point Log

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

Optimizer	- threads	:	20						
Optimizer	- solved problem	:	the primal						
Optimizer	- Constraints	:	1						
Optimizer	- Cones	:	2						
Optimizer	- Scalar variables	:	6	conic	:	6			
Optimizer	- Semi-definite variables:	0	scalarized	:	0				
Factor	- setup time	:	0.00	dense det. time	:	0.00			
Factor	- ML order time	:	0.00	GP order time	:	0.00			
Factor	- nonzeros before factor	:	1	after factor	:	1			
Factor	- dense dim.	:	0	flops	:	1.			
$\hookrightarrow 70e+01$									
ITE	PFEAS	DFEAS	GFEAS	PRSTATUS	POBJ	DOBJ	MU		\hookrightarrow
\hookrightarrow TIME									

(continues on next page)

(continued from previous page)

0	1.0e+00	2.9e-01	3.4e+00	0.00e+00	2.414213562e+00	0.000000000e+00	1.0e+00	┐
↪	0.01							
1	2.7e-01	7.9e-02	2.2e+00	8.83e-01	6.969257574e-01	-9.685901771e-03	2.7e-01	┐
↪	0.01							
2	6.5e-02	1.9e-02	1.2e+00	1.16e+00	7.606090061e-01	6.046141322e-01	6.5e-02	┐
↪	0.01							
3	1.7e-03	5.0e-04	2.2e-01	1.12e+00	7.084385672e-01	7.045122560e-01	1.7e-03	┐
↪	0.01							
4	1.4e-08	4.2e-09	4.9e-08	1.00e+00	7.071067941e-01	7.071067599e-01	1.4e-08	┐
↪	0.01							

The first line displays the number of threads used by the optimizer and the second line indicates if the optimizer chose to solve the primal or dual problem (see *MSK_IPAR_INTPNT_SOLVE_FORM*). The next lines display 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 [Sec. 9.3.1](#) the columns of the iteration log have the following meaning:

- **ITE**: Iteration index k .
- **PFEAS**: $\|Ax^k - b\tau^k\|_\infty$. The numbers in this column should converge monotonically towards zero but may stall at low level due to rounding errors.
- **DFEAS**: $\|A^T y^k + s^k - c\tau^k\|_\infty$. The numbers in this column should converge monotonically towards zero but may stall at low level due to rounding errors.
- **GFEAS**: $|-c^T x^k + b^T y^k - \kappa^k|$. The numbers in this column should converge monotonically towards zero but may stall at low level due to rounding errors.
- **PRSTATUS**: This number converges to 1 if the problem has an optimal solution whereas it converges to -1 if that is not the case.
- **POBJ**: $c^T x^k / \tau^k$. An estimate for the primal objective value.
- **DOBJ**: $b^T y^k / \tau^k$. An estimate for the dual objective value.
- **MU**: $\frac{(x^k)^T s^k + \tau^k \kappa^k}{n+1}$. The numbers in this column should always converge to zero.
- **TIME**: Time spent since the optimization started (in seconds).

9.4 The Optimizer for Mixed-Integer Problems

Solving optimization problems where one or more of the variables are constrained to be integer valued is called Mixed-Integer Optimization (MIO). For an introduction to model building with integer variables, the reader is recommended to consult the **MOSEK Modeling Cookbook**, and for further reading we highlight textbooks such as [\[Wol98\]](#) or [\[CCornuejolsZ14\]](#).

MOSEK can perform mixed-integer

- linear (MILO),
- quadratic (MIQO) and quadratically constrained (MIQCQO), and
- conic (MICO)

optimization, except for mixed-integer semidefinite problems.

By default the mixed-integer optimizer is run-to-run deterministic. This means that if a problem is solved twice on the same computer with identical parameter settings and no time limit, then the obtained solutions will be identical. The mixed-integer optimizer is parallelized, i.e., it can exploit multiple cores during the optimization.

In practice, a predominant special case of integer variables are binary variables, taking values in $\{0, 1\}$. Mixed- or pure binary problems are important subclasses of mixed-integer optimization where all integer variables are of this type. In the general setting however, an integer variable may have arbitrary lower and upper bounds.

9.4.1 Branch-and-Bound

In order to succeed in solving mixed-integer problems, it can be useful to have a basic understanding of the underlying solution algorithms. The most important concept in this regard is arguably the so-called Branch-and-Bound algorithm, employed also by **MOSEK**. In order to comprehend Branch-and-Bound, the concept of a *relaxation* is important.

Consider for example a mixed-integer linear optimization problem of minimization type

$$\begin{aligned} z^* = \quad & \text{minimize} && c^T x \\ & \text{subject to} && Ax = b \\ & && x \geq 0 \\ & && x_j \in \mathbb{Z}, \quad \forall j \in \mathcal{J}. \end{aligned} \quad (9.12)$$

It has the continuous relaxation

$$\begin{aligned} \underline{z} = \quad & \text{minimize} && c^T x \\ & \text{subject to} && Ax = b \\ & && x \geq 0, \end{aligned} \quad (9.13)$$

obtained simply by ignoring the integrality restrictions. The first step in Branch-and-Bound is to solve this so-called *root* relaxation, which is a continuous optimization problem. Since (9.13) is less constrained than (9.12), one certainly gets

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

and \underline{z} is therefore called the *objective bound*: it bounds the optimal objective value from below.

After the solution of the root relaxation, in the most likely outcome there will be one or more integer constrained variables with fractional values, i.e., violating the integrality constraints. Branch-and-Bound now takes such a variable, $x_j = f_j \in \mathbb{R} \setminus \mathbb{Z}$ with $j \in \mathcal{J}$, say, and creates two branches leading to relaxations with the additional constraint $x_j \leq \lfloor f_j \rfloor$ or $x_j \geq \lceil f_j \rceil$, respectively. The intuitive idea here is to push the variable away from the fractional value, closer towards integrality. If the variable was binary, say, branching would lead to fixing its value to 0 in one branch, and to 1 in the other.

The Branch-and-Bound process continues in this way and successively solves relaxations and creates branches to refined relaxations. Whenever a relaxation solution \hat{x} does not violate any integrality constraints, it is feasible to (9.12) and is called an *integer feasible solution*. Clearly, its solution value $\bar{z} := c^T \hat{x}$ is an upper bound on the optimal objective value,

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

Since refining a relaxation by adding constraints to it can only increase its solution value, the objective bound \underline{z} , now defined as the minimum over all solution values of so far solved relaxations, can only increase during the algorithm. If as upper bound \bar{z} one records the solution value of the best integer feasible solution encountered so far, the so-called *incumbent solution*, \bar{z} can only decrease during the algorithm. Since at any time we also have

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

objective bound and incumbent solution value are encapsulating the optimal objective value, eventually converging to it.

The Branch-and-Bound scheme can be depicted by means of a tree, where branches and relaxations correspond to edges and nodes. Figure Fig. 9.1 shows an example of such a tree. The strength of Branch-and-Bound is its ability to prune nodes in this tree, meaning that no new child nodes will be created. Pruning can occur in several cases:

- A relaxation leads to an integer feasible solution \hat{x} . In this case we may update the incumbent and its solution value \bar{z} , but no new branches need to be created.
- A relaxation is infeasible. The subtree rooted at this node cannot contain any feasible relaxation, so it can be discarded.
- A relaxation has a solution value that exceeds \bar{z} . The subtree rooted at this node cannot contain any integer feasible solution with a solution value better than the incumbent we already have, so it can be discarded.

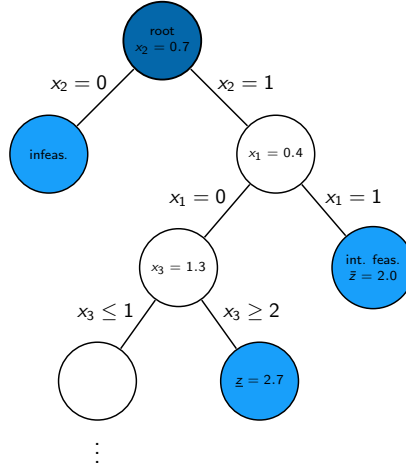


Fig. 9.1: An exemplary Branch-and-Bound tree. Pruned nodes are shown in light blue.

Having objective bound and incumbent solution value is a quite fundamental property of Branch-and-Bound, and helps to assess solution quality and control termination of the algorithm, as we detail in the next section. Note that the above explanation is coined for minimization problems, but the Branch-and-bound scheme has a straightforward extension to maximization problems.

9.4.2 Solution quality and termination criteria

The issue of terminating the mixed-integer optimizer is rather delicate. Recalling the Branch-and-Bound scheme from the previous section, one may see that mixed-integer optimization is generally much harder than continuous optimization; in fact, solving continuous sub-problems is just one component of a mixed-integer optimizer. Despite the ability to prune nodes in the tree, the computational effort required to solve mixed-integer problems grows exponentially with the size of the problem in a worst-case scenario (solving mixed-integer problems is NP-hard). For instance, a problem with n binary variables, may require the solution of 2^n relaxations. The value of 2^n is huge even for moderate values of n . In practice it is often advisable to accept near-optimal or approximate solutions in order to counteract this complexity burden. The user has numerous possibilities of influencing optimizer termination with various parameters, in particular related to solution quality, and the most important ones are highlighted here.

Solution quality in terms of optimality

In order to assess the quality of any incumbent solution in terms of its objective value, one may check the *optimality gap*, defined as

$$\epsilon = |(\text{incumbent solution value}) - (\text{objective bound})| = |\bar{z} - \underline{z}|.$$

It measures how much the objectives of the incumbent and the optimal solution can deviate in the worst case. Often it is more meaningful to look at the *relative optimality gap*

$$\epsilon_{\text{rel}} = \frac{|\bar{z} - \underline{z}|}{\max(\delta_1, |\bar{z}|)}.$$

This is essentially the above *absolute* optimality gap normalized against the magnitude of the incumbent solution value; the purpose of the (small) constant δ_1 is to avoid overweighing incumbent solution values that are very close to zero. The relative optimality gap can thus be interpreted as answering the question: “*Within what fraction of the optimal solution is the incumbent solution in the worst case?*”

Absolute and relative optimality gaps provide useful means to define termination criteria for the mixed-integer optimizer in **MOSEK**. The idea is to terminate the optimization process as soon as the quality of the incumbent solution, measured in absolute or relative gap, is good enough. In fact, whenever an incumbent solution is located, the criterion

$$\bar{z} - \underline{z} \leq \max(\delta_2, \delta_3 \max(\delta_1, |\bar{z}|))$$

is checked. If satisfied, i.e., if either absolute or relative optimality gap are below the thresholds δ_2 or δ_3 , respectively, the optimizer terminates and reports the incumbent as an optimal solution. The optimality gaps can always be retrieved through the information items *MSK_DINF_MIO_OBJ_ABS_GAP* and *MSK_DINF_MIO_OBJ_REL_GAP*.

The tolerances discussed above can be adjusted using suitable parameters, see Table 9.3. By default, the optimality parameters δ_2 and δ_3 are quite small, i.e., restrictive. These default values for the absolute and relative gap amount to solving any instance to (almost) optimality: the incumbent is required to be within at most a tiny percentage of the optimal solution. As anticipated, this is not tractable in most practical situations, and one should resort to finding near-optimal solutions quickly rather than insisting on finding the optimal one. It may happen, for example, that an optimal or close-to-optimal solution is found very early by the optimizer, but it does not terminate because the objective bound \underline{z} is of poor quality. Instead, the vast majority of computational time is spent on trying to improve \underline{z} : a typical situation that practioneers would want to avoid. The concept of optimality gaps is fundamental for controlling solution quality when resorting to near-optimal solutions.

MIO performance tweaks: termination criteria

One of the first things to do in order to cut down excessive solution time is to increase the relative gap tolerance *MSK_DPAR_MIO_TOL_REL_GAP* to some non-default value, so as to not insist on finding optimal solutions. Typical values could be 0.01, 0.05 or 0.1, guaranteeing that the delivered solutions lie within 1%, 5% or 10% of the optimum. Increasing the tolerance will lead to less computational time spent by the optimizer.

Solution quality in terms of feasibility

For an optimizer relying on floating-point arithmetic like the mixed-integer optimizer in **MOSEK**, it may be hard to achieve exact integrality of the solution values of integer variables in most cases, and it makes sense to numerically relax this constraint. Any candidate solution \hat{x} is accepted as integer feasible if the criterion

$$\min(\hat{x}_j - \lfloor \hat{x}_j \rfloor, \lceil \hat{x}_j \rceil - \hat{x}_j) \leq \delta_4 \quad \forall j \in \mathcal{J}$$

is satisfied, meaning that \hat{x}_j is at most δ_4 away from the nearest integer. As above, δ_4 can be adjusted using a parameter, see Table 9.3, and impacts the quality of the acieved solution in terms of integer feasibility. By influencing what solution may be accepted as imcumbent, it can also have an impact on the termination of the optimizer.

MIO performance tweaks: feasibility criteria

Whether increasing the integer feasibility tolerance *MSK_DPAR_MIO_TOL_ABS_RELAX_INT* leads to less solution time is highly problem dependent. Intuitively, the optimizer is more flexible in finding new incumbent soutions so as to improve \bar{z} . But this effect has do be examined with care on individual instances: it may worsen solution quality with no effect at all on the solution time. It may in some cases even lead to contrary effects on the solution time.

Table 9.3: Tolerances for the mixed-integer optimizer.

Tolerance	Parameter name	Default value
δ_1	<i>MSK_DPAR_MIO_REL_GAP_CONST</i>	1.0e-10
δ_2	<i>MSK_DPAR_MIO_TOL_ABS_GAP</i>	0.0
δ_3	<i>MSK_DPAR_MIO_TOL_REL_GAP</i>	1.0e-4
δ_4	<i>MSK_DPAR_MIO_TOL_ABS_RELAX_INT</i>	1.0e-5

Further controlling optimizer termination

There are more ways to limit the computational effort employed by the mixed-integer optimizer by simply limiting the number of explored branches, solved relaxations or updates of the incumbent solution. When any of the imposed limits is hit, the optimizer terminates and the incumbent solution may be retrieved. See Table 9.4 for a list of corresponding parameters. In contrast to the parameters discussed in Sec. 9.4.2, interfering with these does not maintain any guarantees in terms of solution quality.

Table 9.4: Other parameters affecting the integer optimizer termination criterion.

Parameter name	Explanation
<code>MSK_IPAR_MIO_MAX_NUM_BRANCHES</code>	Maximum number of branches allowed.
<code>MSK_IPAR_MIO_MAX_NUM_RELAXS</code>	Maximum number of relaxations allowed.
<code>MSK_IPAR_MIO_MAX_NUM_SOLUTIONS</code>	Maximum number of feasible integer solutions allowed.

9.4.3 Additional components of the mixed-integer Optimizer

The Branch-and-Bound scheme from Sec. 9.4.1 is only the basic skeleton of the mixed-integer optimizer in **MOSEK**, and several components are built on top of that in order to enhance its functionality and increase its speed. A mixed-integer optimizer is sometimes referred to as a “*giant bag of tricks*”, and it would be impossible to describe all of these tricks here. Yet, some of the additional components are worth mentioning to the user. They can be influenced by various user parameters, and although the default values of these parameters are optimized to work well on average mixed-integer problems, it may pay off to adjust them for an individual problem, or a specific problem class.

Presolve

Similar to the case of continuous problems, see Sec. 9.1, the mixed-integer optimizer applies various presolve reductions before the actual solution process is initiated. Just as in the continuous case, the use of presolve can be controlled with the parameter `MSK_IPAR_PREOLVE_USE`.

Primal Heuristics

Solving relaxations in the Branch-and-bound tree to an integer feasible solution \hat{x} is not the only way to find new incumbent solutions. There is a variety of procedures that, given a mixed-integer problem in a generic form like (9.12), attempt to produce integer feasible solutions in an ad-hoc way. These procedures are called Primal Heuristics, and several of them are implemented in **MOSEK**. For example, whenever a relaxation leads to a fractional solution, one may round the solution values of the integer variables, in various ways, and hope that the outcome is still feasible to the remaining constraints. Primal heuristics are mostly employed while processing the root node, but play a role throughout the whole solution process. The goal of a primal heuristic is to improve the incumbent solution and thus the bound \bar{z} , and this can of course affect the quality of the solution that is returned after termination of the optimizer. The user parameters affecting primal heuristics are listed in Table 9.5.

MIO performance tweaks: primal heuristics

- If the mixed-integer optimizer struggles to improve the incumbent solution \bar{z} , see Sec. 9.4.4, it can be helpful to intensify the use of primal heuristics.
 - Set parameters related to primal heuristics to more aggressive values than the default ones, so that more effort is spent in this component. A List of the respective parameters can be found in Table 9.5. In particular, if the optimizer has difficulties finding any integer feasible solution at all, indicated by NA in the column BEST_INT_OBJ in the mixed-integer log, one may try to activate a construction heuristic like the Feasibility Pump with `MSK_IPAR_MIO_FEASPUMP_LEVEL`.
 - Specify a good initial solution: In many cases a good feasible solution is either known or easily computed using problem-specific knowledge that the optimizer does not have. If so, it is usually worthwhile to use this as a starting point for the mixed-integer optimizer.
 - For feasibility problems, i.e., problems having a constant objective, the goal is to find a single integer feasible solution, and this can be hard by itself on some instances. Try setting the

objective to something meaningful anyway, even if the underlying application does not require this. After all, the feasible set is not changed, but the optimizer might benefit from being able to pursue a concrete goal.

- In rare cases it may also happen that the optimizer spends an excessive amount of time on primal heuristics without drawing any benefit from it, and one may try to limit their use with the respective parameters.

Table 9.5: Parameters affecting primal heuristics

Parameter name	Explanation
<i>MSK_IPAR_MIO_HEURISTIC_LEVEL</i>	Primal heuristics aggressivity level.
<i>MSK_IPAR_MIO_RINS_MAX_NODES</i>	Maximum number of nodes allowed in the RINS heuristic.
<i>MSK_IPAR_MIO_FEASPUMP_LEVEL</i>	Way of using the Feasibility Pump heuristic.

Cutting Planes

Cutting planes (cuts) are simply constraints that are valid for a mixed-integer problem, for example in the form (9.12), meaning they do not remove any integer feasible solutions from the feasible set. Therefore they are also called valid inequalities. They do not have to be valid for the relaxation (9.13) though, and of interest and potentially useful are those cuts that do remove solutions from the feasible set of the relaxation. The latter is a superset of the feasible region of the mixed-integer problem, and the rationale behind cuts is thus to bring the integer problem and its relaxation closer together in terms of their feasible sets.

As an example, take the constraints

$$2x_1 + 3x_2 + x_3 \leq 4, \quad x_1, x_2 \in \{0, 1\}, \quad x_3 \geq 0. \quad (9.14)$$

One may realize that there cannot be a feasible solution in which both binary variables take on a value of 1. So certainly

$$x_1 + x_2 \leq 1 \quad (9.15)$$

is a valid inequality. In fact, there is no integer solution satisfying (9.14), but violating (9.15). The latter does cut off a portion of the feasible region of the continuous relaxation of (9.14) though, obtained by replacing $x_1, x_2 \in \{0, 1\}$ with $x_1, x_2 \in [0, 1]$. For example, the fractional point $(x_1, x_2, x_3) = (0.5, 1, 0)$ is feasible to the relaxation, but violates the cut (9.15).

There are many classes of general-purpose cutting planes that may be generated for a mixed-integer problem in a generic form like (9.12), and **MOSEK**'s mixed-integer optimizer supports several of them. For instance, the above is an example of a so-called clique cut. The most effort on generating cutting planes is spent after the solution of the root relaxation, but cuts can also be generated later on in the Branch-and-Bound tree. Cuts aim at improving the objective bound \underline{z} and can thus have significant impact on the solution time. The user parameters affecting cut generation can be seen in Table 9.6.

MIO performance tweaks: cutting planes

- If the mixed-integer optimizer struggles to improve the objective bound \underline{z} , see Sec. 9.4.4, it can be helpful to intensify the use of cutting planes.
 - Some types of cutting planes are not activated by default, but doing so may help to improve the objective bound.
 - The parameters *MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT* and *MSK_IPAR_MIO_CUT_SELECTION_LEVEL* determine how aggressively cuts will be generated and selected.
 - If some valid inequalities can be deduced from problem-specific knowledge that the optimizer does not have, it may be helpful to add these to the problem formulation as constraints. This has to be done with care, since there is a tradeoff between the benefit obtained from an improved objective bound, and the amount of additional constraints that make the relaxations larger.

- In rare cases it may also be observed that the optimizer spends an excessive amount of time on cutting planes, see [Sec. 9.4.4](#), and one may limit their use with `MSK_IPAR_MIO_MAX_NUM_ROOT_CUT_ROUNDS`, or by disabling a certain type of cutting planes.

Table 9.6: Parameters affecting cutting planes

Parameter name	Explanation
<code>MSK_IPAR_MIO_CUT_CLIQUE</code>	Should clique cuts be enabled?
<code>MSK_IPAR_MIO_CUT_CMIR</code>	Should mixed-integer rounding cuts be enabled?
<code>MSK_IPAR_MIO_CUT_GMI</code>	Should GMI cuts be enabled?
<code>MSK_IPAR_MIO_CUT_IMPLIED_BOUND</code>	Should implied bound cuts be enabled?
<code>MSK_IPAR_MIO_CUT_KNAPSACK_COVER</code>	Should knapsack cover cuts be enabled?
<code>MSK_IPAR_MIO_CUT_LIPRO</code>	Should lift-and-project cuts be enabled?
<code>MSK_IPAR_MIO_CUT_SELECTION_LEVEL</code>	Cut selection aggressivity level.
<code>MSK_IPAR_MIO_MAX_NUM_ROOT_CUT_ROUNDS</code>	Maximum number of root cut rounds.
<code>MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT</code>	Minimum required objective bound improvement during root cut generation.

9.4.4 The Mixed-Integer Log

Below is a typical log output from the mixed-integer optimizer:

Presolved problem: 1176 variables, 1344 constraints, 4968 non-zeros						
Presolved problem: 328 general integer, 392 binary, 456 continuous						
Clique table size: 55						
BRANCHES	RELAXS	ACT_NDS	DEPTH	BEST_INT_OBJ	BEST_RELAX_OBJ	REL_GAP(
→%)	TIME					
0	0	1	0	8.3888091139e+07	NA	NA
→	0.0					
0	1	1	0	8.3888091139e+07	2.5492512136e+07	69.61
→	0.1					
0	1	1	0	3.1273162420e+07	2.5492512136e+07	18.48
→	0.1					
0	1	1	0	2.6047699632e+07	2.5492512136e+07	2.13
→	0.2					
Cut generation started.						
0	1	1	0	2.6047699632e+07	2.5492512136e+07	2.13
→	0.2					
0	2	1	0	2.6047699632e+07	2.5589986247e+07	1.76
→	0.2					
Cut generation terminated. Time = 0.05						
0	4	1	0	2.5990071367e+07	2.5662741991e+07	1.26
→	0.3					
0	8	1	0	2.5971002767e+07	2.5662741991e+07	1.19
→	0.5					
0	11	1	0	2.5925040617e+07	2.5662741991e+07	1.01
→	0.5					
0	12	1	0	2.5915504014e+07	2.5662741991e+07	0.98
→	0.5					
2	23	1	0	2.5915504014e+07	2.5662741991e+07	0.98
→	0.6					
14	35	1	0	2.5915504014e+07	2.5662741991e+07	0.98
→	0.6					
[...]						

(continues on next page)

```

Objective of best integer solution : 2.578282162804e+07
Best objective bound               : 2.569877601306e+07
Construct solution objective       : Not employed
User objective cut value           : Not employed
Number of cuts generated           : 192
  Number of Gomory cuts             : 52
  Number of CMIR cuts               : 137
  Number of clique cuts             : 3
Number of branches                 : 29252
Number of relaxations solved        : 31280
Number of interior point iterations: 16
Number of simplex iterations        : 105440
Time spend presolving the root      : 0.03
Time spend optimizing the root      : 0.07
Mixed integer optimizer terminated. Time: 6.46

```

The first lines contain a summary of the problem after mixed-integer presolve has been applied. This is followed by the iteration log, reflecting the progress made during the Branch-and-bound process. The columns have the following meanings:

- **BRANCHES**: Number of branches / nodes generated.
- **RELAXS**: Number of relaxations solved.
- **ACT_NDS**: Number of active / non-processed nodes.
- **DEPTH**: Depth of the last solved node.
- **BEST_INT_OBJ**: The incumbent solution / best integer objective value, \bar{z} .
- **BEST_RELAX_OBJ**: The objective bound, \underline{z} .
- **REL_GAP(%)**: Relative optimality gap, $100\% \cdot \epsilon_{\text{rel}}$
- **TIME**: Time (in seconds) from the start of optimization.

The beginning and the end of the root cut generation is highlighted as well, and the number of log lines in between reflects to the computational effort spent here.

Finally there is a summary of the optimization process, containing also information on the type of generated cuts and the total number of iterations needed to solve all occurring continuous relaxations.

When the solution time for a mixed-integer problem has to be cut down, it can sometimes be useful to examine the log in order to understand where time is spent and what might be improved. In particular, it might happen that the values in either of the columns **BEST_INT_OBJ** or **BEST_RELAX_OBJ** stall over a long period of log lines, an indication that the optimizer has a hard time improving either the incumbent solution, i.e., \bar{z} , or the objective bound \underline{z} , see also [Sec. 9.4.3](#) and [Sec. 9.4.3](#).

9.4.5 Mixed-Integer Nonlinear Optimization

Due to the involved non-linearities, MI(QC)QO or MICO problems are on average harder than MILO problems of comparable size. Yet, the Branch-and-Bound scheme can be applied to these problem classes in a straightforward manner. The relaxations have to be solved as conic problems with the interior point algorithm in that case, see [Sec. 9.3](#), opposed to MILO where it is often beneficial to solve relaxations with the dual simplex method, see [Sec. 9.2.3](#). There is another solution approach for these types of problems implemented in **MOSEK**, namely the Outer-Approximation algorithm, making use of dynamically refined linear approximations of the non-linearities.

MICO performance tweaks: choice of algorithm

Whether conic Branch-and-Bound or Outer-Approximation is applied to a mixed-integer conic problem can be set with `MSK_IPAR_MIO_CONIC_OUTER_APPROXIMATION`. The best value for this option is highly problem dependent.

MI(QC)QO

MOSEK is specialized in solving linear and conic optimization problems, both with or without mixed-integer variables. Just like for continuous problems, mixed-integer quadratic problems are converted internally to conic form, see [Sec. 8.2.4](#)

Contrary to the continuous case, **MOSEK** can solve certain mixed-integer quadratic problems where one or more of the involved matrices are not positive semidefinite, so-called non-convex MI(QC)QO problems. These are automatically reformulated to an equivalent convex MI(QC)QO problem, provided that such a reformulation is possible on the given instance (otherwise **MOSEK** will reject the problem and issue an error message). The concept of reformulations can also affect the solution times of MI(QC)QO problems.

MI(QC)QO performance tweaks: applying a reformulation method

There are several reformulation methods for MI(QC)QO problems, available through the parameter `MSK_IPAR_MIO_QCQO_REFORMULATION_METHOD`. The chosen method can have significant impact on the mixed-integer optimizer's speed on such problems, both convex and non-convex. The best value for this option is highly problem dependent.

9.4.6 Disjunctive constraints

Problems with disjunctive constraints (DJC) are typically reformulated to mixed-integer problems, and even if this is not the case they are solved with an algorithm that is based on the mixed-integer optimizer. In **MOSEK**, these problems thus fall into the realm of MIO. In particular, **MOSEK** automatically attempts to replace any DJC by so called big-M constraints, potentially after transforming it to several, less complicated DJCs. As an example, take the DJC

$$[z = 0] \vee [z = 1, x_1 + x_2 \geq 1000],$$

where $z \in \{0, 1\}$ and $x_1, x_2 \in [0, 750]$. This is an example of a DJC formulation of a so-called indicator constraint. A big-M reformulation is given by

$$x_1 + x_2 \geq 1000 - M \cdot (1 - z),$$

where $M > 0$ is a large constant. The practical difficulty of these constructs is that M should always be sufficiently large, but ideally not larger. Too large values for M can be harmful for the mixed-integer optimizer. During presolve, and taking into account the bounds of the involved variables, **MOSEK** automatically reformulates DJCs to big-M constraints if the required M values do not exceed the parameter `MSK_DPAR_MIO_DJC_MAX_BIGM`. From a performance point-of-view, all DJCs would ideally be linearized to big-Ms after presolve without changing this parameter's default value of 1.0e6. Whether or not this is the case can be seen by retrieving the information item `MSK_IINF_MIO_PRE SOLVED_NUMDJC`, or by a line in the mixed-integer optimizer's log as in the example below. Both state the number of remaining disjunctions after presolve.

Presolved problem: 305 variables, 204 constraints, 708 non-zeros						
Presolved problem: 0 general integer, 100 binary, 205 continuous						
Presolved problem: 100 disjunctions						
Clique table size: 0						
BRANCHES	RELAXS	ACT_NDS	DEPTH	BEST_INT_OBJ	BEST_RELAX_OBJ	REL_GAP(
→ %)	TIME					
0	1	1	0	NA	0.0000000000e+00	NA
→	0.0					
0	1	1	0	5.0574653969e+05	0.0000000000e+00	100.00
→	0.0					
[...]						

DJC performance tweaks: managing variable bounds

- Always specify the tightest known bounds on the variables of any problem with DJCs, even if they seem trivial from the user-perspective. The mixed-integer optimizer can only benefit from these when reformulating DJCs and thus gain performance; even if bounds don't help with reformulations, it is very unlikely that they hurt the optimizer.
- Increasing `MSK_DPAR_MIO_DJC_MAX_BIGM` can lead to more DJC reformulations and thus increase optimizer speed, but it may in turn hurt numerical solution quality and has to be examined with care. The other way round, on numerically challenging instances with DJCs, decreasing `MSK_DPAR_MIO_DJC_MAX_BIGM` may lead to numerically more robust solutions.

9.4.7 Randomization

A mixed-integer optimizer is usually prone to performance variability, meaning that a small change in either

- problem data, or
- computer hardware, or
- algorithmic parameters

can lead to significant changes in solution time, due to different solution paths in the Branch-and-Bound tree. In extreme cases the exact same problem can vary from being solvable in less than a second to seemingly unsolvable in any reasonable amount of time on a different computer.

One practical implication of this is that one should ideally verify whether a seemingly beneficial set of parameters, established experimentally on a single problem, is still beneficial (on average) on a larger set of problems from the same problem class. This protects against making parameter changes that had positive effects only due to random effects on that single problem.

In the absence of a large set of test problems, one may also change the random seed of the optimizer to a series of different values in order to hedge against drawing such wrong conclusions regarding parameters. The random seed, accessible through `MSK_IPAR_MIO_SEED`, impacts for example random tie-breaking in many of the mixed-integer optimizer's components. Changing the random seed can be combined with a permutation of the problem data to further incite randomness, accessible through the parameter `MSK_IPAR_MIO_DATA_PERMUTATION_METHOD`.

9.4.8 Further performance tweaks

In addition to what was mentioned previously, there may be other ways to speed up the solution of a given mixed-integer problem. For example, there are further user parameters affecting some algorithmic settings in the mixed-integer optimizer. As mentioned above, default parameter values are optimized to work well on average, but on individual problems they may be adjusted.

MIO performance tweaks: miscellaneous

- When relaxations in the the Branch-and-Bound tree are linear optimization problems (e.g., in MILO or when solving MICO problems with the Outer-Approximation method), it is usually best to employ the dual simplex method for their solution. In rare cases the primal simplex method may actually be the better choice, and this can be set with the parameter `MSK_IPAR_MIO_NODE_OPTIMIZER`.
- Some problems are numerically more challenging than others, for example if the ratio between the smallest and the largest involved coefficients is large, say $\geq 1e9$. An indication of numerical issues are, for example, large violations in the final solution, observable in the solution summary of the log output, see [Sec. 7.1.3](#). Similarly, a problem that is known to be feasible by the user may be declared infeasible by the optimizer. In such cases it is usually best to try to rescale the model. Otherwise, the mixed-integer optimizer can be instructed to be more cautious regarding numerics with the parameter `MSK_IPAR_MIO_NUMERICAL_EMPHASIS_LEVEL`. This may in turn be at the cost of solution speed though.
- Improve the formulation: A MIO 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 [\[Wol98\]](#).

Chapter 10

Additional features

In this section we describe additional features and tools which enable more detailed analysis of optimization problems with **MOSEK**.

10.1 Problem Analyzer

The problem analyzer prints a survey of the structure of the problem, with information about linear constraints and objective, quadratic constraints, conic constraints and variables.

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

The problem analyzer is run from the command line using the `-anapro` argument and produces output similar to the following (this is the problem analyzer's survey of the `aflow30a` problem from the MIPLIB 2003 collection.)

Analyzing the problem					
*** Structural report					
Dimensions					
	Constraints	Variables	Matrix var.	Cones	
	479	842	0	0	
Constraint and bound types					
	Free	Lower	Upper	Ranged	Fixed
Constraints:	0	0	421	0	58
Variables:	0	0	0	842	0
Integer constraint types					
	Binary	General			
	421	0			
*** Data report					
	Nonzeros	Min	Max		
cj :	421	1.1e+01	5.0e+02		
Aij :	2091	1.0e+00	1.0e+02		
	# finite	Min	Max		
blci :	58	1.0e+00	1.0e+01		
buci :	479	0.0e+00	1.0e+01		
blxj :	842	0.0e+00	0.0e+00		
buxj :	842	1.0e+00	1.0e+02		

(continues on next page)

*** Done analyzing the problem

The survey is divided into a structural and numerical report. The content should be self-explanatory.

10.2 Automatic Repair of Infeasible Problems

MOSEK provides an automatic repair tool for infeasible linear problems which we cover in this section. Note that most infeasible models are so due to bugs which can (and should) be more reliably fixed manually, using the knowledge of the model structure. We discuss this approach in [Sec. 7.3](#).

10.2.1 Automatic repair

The main idea can be described as follows. Consider the linear optimization problem with m constraints and n variables

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

which is assumed to be infeasible.

One way of making the problem feasible is to reduce the lower bounds and increase the upper bounds. If the change is sufficiently large the problem becomes feasible. Now an obvious idea is to compute the optimal relaxation by solving an optimization problem. The problem

$$\begin{array}{ll} \text{minimize} & p(v_l^c, v_u^c, v_l^x, v_u^x) \\ \text{subject to} & l^c - v_l^c \leq Ax \leq u^c + v_u^c, \\ & l^x - v_l^x \leq x \leq u^x + v_u^x, \\ & v_l^c, v_u^c, v_l^x, v_u^x \geq 0 \end{array} \quad (10.1)$$

does exactly that. The additional variables $(v_l^c)_i$, $(v_u^c)_i$, $(v_l^x)_j$ and $(v_u^x)_j$ are *elasticity* variables because they allow a constraint to be violated and hence add some elasticity to the problem. For instance, the elasticity variable $(v_l^c)_i$ controls how much the lower bound $(l^c)_i$ should be relaxed to make the problem feasible. Finally, the so-called penalty function

$$p(v_l^c, v_u^c, v_l^x, v_u^x)$$

is chosen so it penalizes changes to bounds. Given the weights

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

a natural choice is

$$p(v_l^c, v_u^c, v_l^x, v_u^x) = (w_l^c)^T v_l^c + (w_u^c)^T v_u^c + (w_l^x)^T v_l^x + (w_u^x)^T v_u^x.$$

Hence, the penalty function $p()$ is a weighted sum of the elasticity variables and therefore the problem (10.1) keeps the amount of relaxation at a minimum. Please observe that

- the problem (10.1) is always feasible.
- a negative weight implies problem (10.1) is unbounded. For this reason if the value of a weight is negative **MOSEK** fixes the associated elasticity variable to zero. Clearly, if one or more of the weights are negative, it may imply that it is not possible to repair the problem.

A simple choice of weights is to set them all to 1, but of course that does not take into account that constraints may have different importance.

Caveats

Observe if the infeasible problem

$$\begin{array}{lll} \text{minimize} & x + z \\ \text{subject to} & x = -1, \\ & x \geq 0 \end{array}$$

is repaired then it will become unbounded. Hence, a repaired problem may not have an optimal solution.

Another and more important caveat is that only a minimal repair is performed i.e. the repair that barely makes the problem feasible. Hence, the repaired problem is barely feasible and that sometimes makes the repaired problem hard to solve.

Using the automatic repair tool

In this subsection we consider an infeasible linear optimization example:

$$\begin{array}{llll} \text{minimize} & -10x_1 & -9x_2, \\ \text{subject to} & 7/10x_1 + 1x_2 \leq 630, \\ & 1/2x_1 + 5/6x_2 \leq 600, \\ & 1x_1 + 2/3x_2 \leq 708, \\ & 1/10x_1 + 1/4x_2 \leq 135, \\ & x_1, & x_2 \geq 0, \\ & & x_2 \geq 650. \end{array} \quad (10.2)$$

The problem (10.2) is contained in a file:

Listing 10.1: Problem (10.2) in LP format.

```
minimize
  obj: - 10 x1 - 9 x2
st
  c1: + 7e-01 x1 + x2 <= 630
  c2: + 5e-01 x1 + 8.333333333e-01 x2 <= 600
  c3: + x1 + 6.6666667e-01 x2 <= 708
  c4: + 1e-01 x1 + 2.5e-01 x2 <= 135
bounds
  x2 >= 650
end
```

Given the assumption that all weights are 1 the command

```
mosek -primalrepair -d MSK_IPAR_LOG_FEAS_REPAIR 3 feasrepair.lp
```

will form the repaired problem and solve it. The parameter `MSK_IPAR_LOG_FEAS_REPAIR` controls the amount of log output from the repair. A value of 2 causes the optimal repair to be printed out. The output from running the above command is:

```
MOSEK Version 9.0.0.25(ALPHA) (Build date: 2017-11-7 16:11:50)
Copyright (c) MOSEK ApS, Denmark. WWW: mosek.com
Platform: Linux/64-X86

Open file 'feasrepair.lp'
Reading started.
Reading terminated. Time: 0.00

Read summary
  Type           : LO (linear optimization problem)
  Objective sense : min
  Scalar variables : 2
  Matrix variables : 0
```

(continues on next page)

```

Constraints      : 4
Cones            : 0
Time             : 0.0

```

Problem

```

Name            :
Objective sense  : min
Type            : LO (linear optimization problem)
Constraints      : 4
Cones           : 0
Scalar variables : 2
Matrix variables : 0
Integer variables : 0

```

Primal feasibility repair started.

Optimizer started.

Presolve started.

Linear dependency checker started.

Linear dependency checker terminated.

Eliminator started.

Freed constraints in eliminator : 2

Eliminator terminated.

```

Eliminator - tries           : 1                time           : 0.00
Lin. dep.  - tries           : 1                time           : 0.00
Lin. dep.  - number          : 0

```

Presolve terminated. Time: 0.00

Problem

```

Name            :
Objective sense  : min
Type            : LO (linear optimization problem)
Constraints      : 8
Cones           : 0
Scalar variables : 14
Matrix variables : 0
Integer variables : 0

```

```

Optimizer - threads           : 20
Optimizer - solved problem    : the primal
Optimizer - Constraints        : 2
Optimizer - Cones             : 0
Optimizer - Scalar variables   : 5                conic           : 0
Optimizer - Semi-definite variables: 0            scalarized        : 0
Factor - setup time           : 0.00              dense det. time   : 0.00
Factor - ML order time        : 0.00              GP order time     : 0.00
Factor - nonzeros before factor : 3                after factor      : 3
Factor - dense dim.           : 0                flops             : 5.

```

→ 00e+01

	ITE	PFEAS	DFEAS	GFEAS	PRSTATUS	POBJ	DOBJ	MU
→ TIME								
0	2.7e+01	1.0e+00	4.0e+00	1.00e+00	3.000000000e+00	0.000000000e+00	1.0e+00	→
→ 0.00								
1	2.5e+01	9.1e-01	1.4e+00	0.00e+00	8.711262850e+00	1.115287830e+01	2.4e+00	→
→ 0.00								
2	2.4e+00	8.8e-02	1.4e-01	-7.33e-01	4.062505701e+01	4.422203730e+01	2.3e-01	→
→ 0.00								
3	9.4e-02	3.4e-03	5.5e-03	1.33e+00	4.250700434e+01	4.258548510e+01	9.1e-03	→
→ 0.00								

→ 0.00

→ 0.00

→ 0.00

→ 0.00

→ 0.00

→ 0.00

→ 0.00

→ 0.00

→ 0.00

→ 0.00

(continues on next page)

(continued from previous page)

```
4  2.0e-05  7.2e-07  1.1e-06  1.02e+00  4.249996599e+01  4.249998669e+01  1.9e-06┐
↳ 0.00
5  2.0e-09  7.2e-11  1.1e-10  1.00e+00  4.250000000e+01  4.250000000e+01  1.9e-10┐
↳ 0.00
Basis identification started.
Basis identification terminated. Time: 0.00
Optimizer terminated. Time: 0.01

Basic solution summary
  Problem status : PRIMAL_AND_DUAL_FEASIBLE
  Solution status : OPTIMAL
  Primal.  obj: 4.250000000e+01   nrm: 6e+02   Viol.  con: 1e-13   var: 0e+00
  Dual.    obj: 4.249999999e+01   nrm: 2e+00   Viol.  con: 0e+00   var: 9e-11
Optimal objective value of the penalty problem: 4.250000000000e+01

Repairing bounds.
Increasing the upper bound 1.35e+02 on constraint 'c4' (3) with 2.25e+01.
Decreasing the lower bound 6.50e+02 on variable 'x2' (4) with 2.00e+01.
Primal feasibility repair terminated.
Optimizer started.
Optimizer terminated. Time: 0.00

Interior-point solution summary
  Problem status : PRIMAL_AND_DUAL_FEASIBLE
  Solution status : OPTIMAL
  Primal.  obj: -5.670000000e+03   nrm: 6e+02   Viol.  con: 0e+00   var: 0e+00
  Dual.    obj: -5.670000000e+03   nrm: 1e+01   Viol.  con: 0e+00   var: 0e+00

Basic solution summary
  Problem status : PRIMAL_AND_DUAL_FEASIBLE
  Solution status : OPTIMAL
  Primal.  obj: -5.670000000e+03   nrm: 6e+02   Viol.  con: 0e+00   var: 0e+00
  Dual.    obj: -5.670000000e+03   nrm: 1e+01   Viol.  con: 0e+00   var: 0e+00

Optimizer summary
Optimizer                -                time: 0.00
  Interior-point          - iterations : 0      time: 0.00
    Basis identification -                time: 0.00
      Primal              - iterations : 0      time: 0.00
      Dual                - iterations : 0      time: 0.00
      Clean primal        - iterations : 0      time: 0.00
      Clean dual          - iterations : 0      time: 0.00
    Simplex               -                time: 0.00
      Primal simplex      - iterations : 0      time: 0.00
      Dual simplex        - iterations : 0      time: 0.00
    Mixed integer         - relaxations: 0      time: 0.00
```

In this case the optimal repair it is to increase the upper bound on constraint c4 by 22.5 and decrease the lower bound on variable x2 by 20.

10.3 Sensitivity Analysis

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

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

References

The book [Chvatal83] discusses the classical sensitivity analysis in Chapter 10 whereas the book [RTV97] presents a modern introduction to sensitivity analysis. Finally, it is recommended to read the short paper [Wal00] to avoid some of the pitfalls associated with sensitivity analysis.

Warning: Currently, sensitivity analysis is only available for continuous linear optimization problems. Moreover, **MOSEK** can only deal with perturbations of bounds and objective function coefficients.

10.3.1 Sensitivity Analysis for Linear Problems

The Optimal Objective Value Function

Assume that we are given the problem

$$\begin{aligned} z(l^c, u^c, l^x, u^x, c) = & \text{minimize} && c^T x \\ & \text{subject to} && l^c \leq Ax \leq u^c, \\ & && l^x \leq x \leq u^x, \end{aligned} \quad (10.3)$$

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

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

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

$$f_{l_i^c}(\beta) \quad (10.4)$$

shows the optimal objective value as a function of β . Please note that a change in β corresponds to a perturbation in l_i^c and hence (10.4) shows the optimal objective value as a function of varying l_i^c with the other bounds fixed.

It is possible to prove that the function (10.4) is a piecewise linear and convex function, i.e. its graph may look like in Fig. 10.1 and Fig. 10.2.

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

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

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

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

for which

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

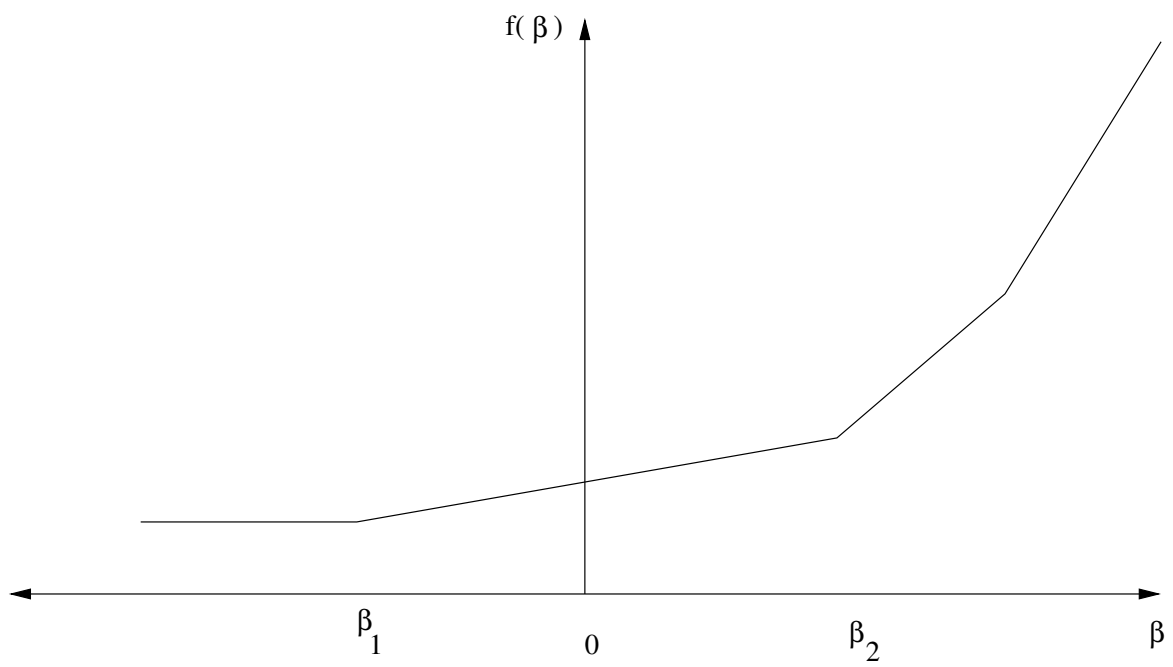


Fig. 10.1: $\beta = 0$ is in the interior of linearity interval.

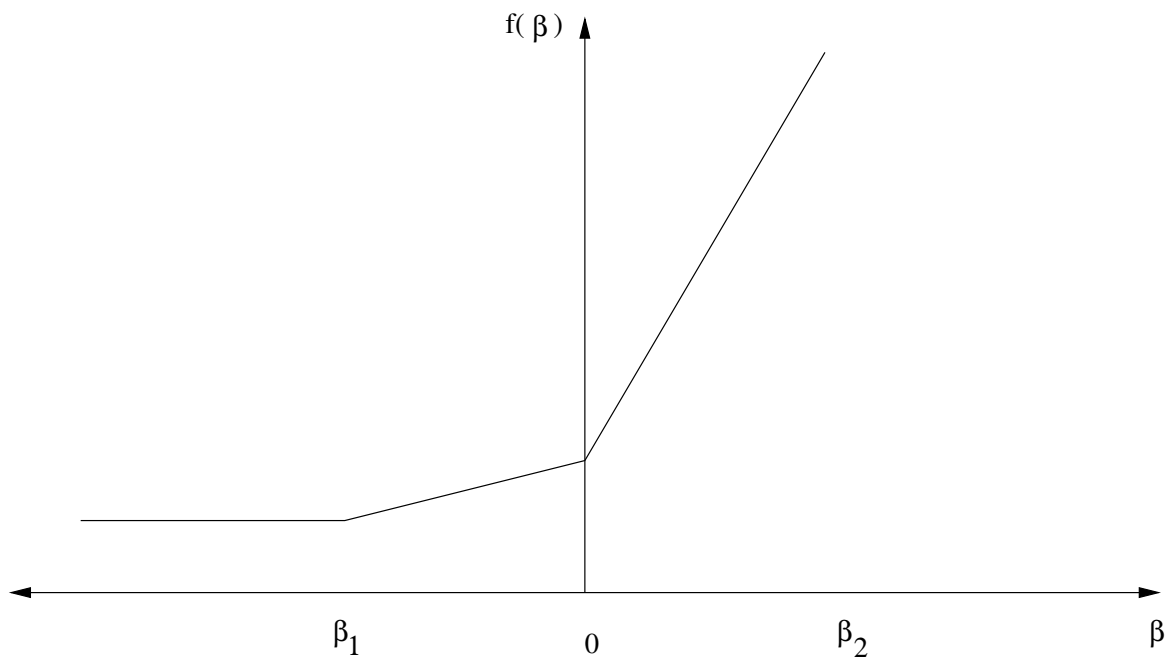


Fig. 10.2: $\beta = 0$ is a breakpoint.

Since $f_{l_i^c}$ is not a smooth function $f'_{l_i^c}$ may not be defined at 0, as illustrated in Fig. 10.2. In this case we can define a left and a right shadow price and a left and a right linearity interval.

The function $f_{l_i^c}$ considered only changes in l_i^c . We can define similar functions for the remaining parameters of the z defined in (10.3) as well:

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

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

Equality Constraints

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

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

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

The Basis Type Sensitivity Analysis

The classical sensitivity analysis discussed in most textbooks about linear optimization, e.g. [Chvatal83], is based on an optimal basis. This method may produce misleading results [RTV97] but is computationally cheap. This is the type of sensitivity analysis implemented in **MOSEK**.

We will now briefly discuss the basis type sensitivity analysis. Given an optimal basic solution which provides a partition of variables into basic and non-basic variables, the basis type sensitivity analysis computes the linearity interval $[\beta_1, \beta_2]$ so that the basis remains optimal for the perturbed problem. A shadow price associated with the linearity interval is also computed. However, it is well-known that an optimal basic solution may not be unique and therefore the result depends on the optimal basic solution employed in the sensitivity analysis. If the optimal objective value function has a breakpoint for $\beta = 0$ then 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.

Example: Sensitivity Analysis

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 Fig. 10.3.

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

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

subject to

$$\begin{aligned} x_{11} + x_{12} &\leq 400, \\ x_{23} + x_{24} &\leq 1200, \\ x_{31} + x_{33} + x_{34} &\leq 1000, \\ x_{11} + x_{31} &= 800, \\ x_{12} + x_{33} &= 100, \\ x_{23} + x_{33} &= 500, \\ x_{24} + x_{34} &= 500, \\ x_{11}, x_{12}, x_{23}, x_{24}, x_{31}, x_{33}, x_{34} &\geq 0. \end{aligned} \tag{10.5}$$

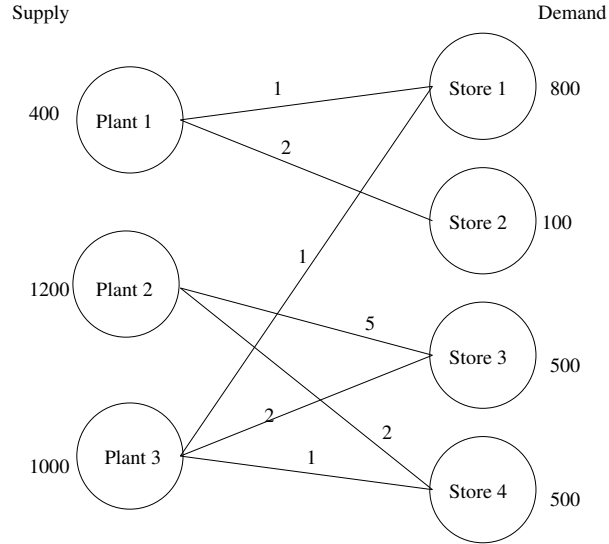


Fig. 10.3: Supply, demand and cost of transportation.

The sensitivity parameters are shown in Table 10.1 and Table 10.2.

Table 10.1: Ranges and shadow prices related to bounds on constraints and variables.

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

Table 10.2: Ranges and shadow prices related to the objective coefficients.

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

Examining the results from the sensitivity analysis we see that for constraint number 1 we have $\sigma_1 = 3$ and $\beta_1 = -300$, $\beta_2 = 0$.

If the upper bound on constraint 1 is decreased by

$$\beta \in [0, 300]$$

then the optimal objective value will increase by the value

$$\sigma_1 \beta = 3\beta.$$

10.3.2 Sensitivity Analysis with MOSEK

A sensitivity analysis can be performed with the **MOSEK** command line tool specifying the option `-sen`, e.g.

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

where `sensitivity.ssp` is a file in the format described in the next section. The `ssp` file describes which parts of the problem the sensitivity analysis should be performed on, see [Sec. 10.3.2](#).

By default results are written to a file named `myproblem.sen`. If necessary, this file name can be changed by setting the `MSK_SPAR_SENSITIVITY_RES_FILE_NAME` parameter.

Sensitivity Analysis Specification File

MOSEK employs an MPS-like file format to specify on which model parameters the sensitivity analysis should be performed. The format of the sensitivity specification file is shown in [Listing 10.2](#), where capitalized names are keywords, and names in brackets are names of the constraints and variables to be included in the analysis.

Listing 10.2: Sensitivity analysis file specification.

```
BOUNDS CONSTRAINTS
U|L|LU [cname1]
U|L|LU [cname2] - [cname3]
BOUNDS VARIABLES
U|L|LU [vname1]
U|L|LU [vname2] - [vname3]
OBJECTIVE VARIABLES
[vname1]
[vname2] - [vname3]
```

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

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

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

Recall from [Sec. 10.3.1](#) that equality constraints are handled in a special way. Sensitivity analysis of an equality constraint can be specified with either L, U, or LU, all indicating the same, namely that upper and lower bounds (which are equal) are perturbed simultaneously.

As an example consider

BOUNDS CONSTRAINTS

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

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

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

BOUNDS CONSTRAINTS

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

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

Example: Sensitivity Analysis from Command Line

As an example consider problem (10.5): the sensitivity file shown below (included in the distribution among the examples).

Listing 10.3: Sensitivity file for problem (10.5).

```
* Comment 1

BOUNDS CONSTRAINTS
U "c1"          * Analyze upper bound for constraints named c1
U 2             * Analyze upper bound for constraints with index 2
U 3-5           * Analyze upper bound for constraint with index in interval [3:5]

VARIABLES CONSTRAINTS
L 2-4           * This section specifies which bounds on variables should be analyzed.
↳
L "x11"

OBJECTIVE CONSTRAINTS
"x11"           * This section specifies which objective coefficients should be
↳analysed.
2
```

The command

```
mosek transport.lp -sen sensitivity.ssp
```

produces the output file as follow

Listing 10.4: Results of sensitivity analysis

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

(continues on next page)

(continued from previous page)

BOUNDS VARIABLES					
INDEX	NAME	BOUND	LEFTRANGE	RIGHTRANGE	LEFTPRICE
	↪RIGHTPRICE				
2	x23	LO	-6.574875e-18	5.000000e+02	2.000000e+00
	↪2.000000e+00				
3	x24	LO	-inf	5.000000e+02	0.000000e+00
	↪0.000000e+00				
4	x31	LO	-inf	5.000000e+02	0.000000e+00
	↪0.000000e+00				
0	x11	LO	-inf	3.000000e+02	0.000000e+00
	↪0.000000e+00				
OBJECTIVE VARIABLES					
INDEX	NAME		LEFTRANGE	RIGHTRANGE	LEFTPRICE
	↪RIGHTPRICE				
0	x11		-inf	1.000000e+00	3.000000e+02
	↪3.000000e+02				
2	x23		-2.000000e+00	+inf	0.000000e+00
	↪0.000000e+00				

Controlling Log Output

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

The parameter *MSK_IPAR_LOG_SENSITIVITY_OPT* controls the amount of debug information on internal calculations from the sensitivity analysis.

Chapter 11

API Reference

- **Optimizer parameters:**
 - *Double, Integer, String*
 - *Full list*
 - *Browse by topic*
- *Optimizer response codes*
- *Constants*
- *List of supported domains*

11.1 Parameters grouped by topic

Analysis

- *MSK_DPAR_ANA_SOL_INFEAS_TOL*
- *MSK_IPAR_ANA_SOL_BASIS*
- *MSK_IPAR_ANA_SOL_PRINT_VIOLATED*
- *MSK_IPAR_LOG_ANA_PRO*

Basis identification

- *MSK_DPAR_SIM_LU_TOL_REL_PIV*
- *MSK_IPAR_BI_CLEAN_OPTIMIZER*
- *MSK_IPAR_BI_IGNORE_MAX_ITER*
- *MSK_IPAR_BI_IGNORE_NUM_ERROR*
- *MSK_IPAR_BI_MAX_ITERATIONS*
- *MSK_IPAR_INTPNT_BASIS*
- *MSK_IPAR_LOG_BI*
- *MSK_IPAR_LOG_BI_FREQ*

Conic interior-point method

- *MSK_DPAR_INTPNT_CO_TOL_DFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_INFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_MU_RED*
- *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_CO_TOL_PFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_REL_GAP*

Data check

- *MSK_DPAR_DATA_SYM_MAT_TOL*
- *MSK_DPAR_DATA_SYM_MAT_TOL_HUGE*
- *MSK_DPAR_DATA_SYM_MAT_TOL_LARGE*
- *MSK_DPAR_DATA_TOL_AIJ_HUGE*
- *MSK_DPAR_DATA_TOL_AIJ_LARGE*
- *MSK_DPAR_DATA_TOL_BOUND_INF*
- *MSK_DPAR_DATA_TOL_BOUND_WRN*
- *MSK_DPAR_DATA_TOL_C_HUGE*
- *MSK_DPAR_DATA_TOL_CJ_LARGE*
- *MSK_DPAR_DATA_TOL_QIJ*
- *MSK_DPAR_DATA_TOL_X*
- *MSK_DPAR_SEMIDEFINITE_TOL_APPROX*
- *MSK_IPAR_CHECK_CONVEXITY*
- *MSK_IPAR_LOG_CHECK_CONVEXITY*

Data input/output

- *MSK_IPAR_INFEAS_REPORT_AUTO*
- *MSK_IPAR_LOG_FILE*
- *MSK_IPAR_OPF_WRITE_HEADER*
- *MSK_IPAR_OPF_WRITE_HINTS*
- *MSK_IPAR_OPF_WRITE_LINE_LENGTH*
- *MSK_IPAR_OPF_WRITE_PARAMETERS*
- *MSK_IPAR_OPF_WRITE_PROBLEM*
- *MSK_IPAR_OPF_WRITE_SOL_BAS*
- *MSK_IPAR_OPF_WRITE_SOL_ITG*
- *MSK_IPAR_OPF_WRITE_SOL_ITR*
- *MSK_IPAR_OPF_WRITE_SOLUTIONS*

- *MSK_IPAR_PARAM_READ_CASE_NAME*
- *MSK_IPAR_PARAM_READ_IGN_ERROR*
- *MSK_IPAR_PTF_WRITE_PARAMETERS*
- *MSK_IPAR_PTF_WRITE_SOLUTIONS*
- *MSK_IPAR_PTF_WRITE_TRANSFORM*
- *MSK_IPAR_READ_DEBUG*
- *MSK_IPAR_READ_KEEP_FREE_CON*
- *MSK_IPAR_READ_MPS_FORMAT*
- *MSK_IPAR_READ_MPS_WIDTH*
- *MSK_IPAR_READ_TASK_IGNORE_PARAM*
- *MSK_IPAR_SOL_READ_NAME_WIDTH*
- *MSK_IPAR_SOL_READ_WIDTH*
- *MSK_IPAR_WRITE_BAS_CONSTRAINTS*
- *MSK_IPAR_WRITE_BAS_HEAD*
- *MSK_IPAR_WRITE_BAS_VARIABLES*
- *MSK_IPAR_WRITE_COMPRESSION*
- *MSK_IPAR_WRITE_DATA_PARAM*
- *MSK_IPAR_WRITE_FREE_CON*
- *MSK_IPAR_WRITE_GENERIC_NAMES*
- *MSK_IPAR_WRITE_GENERIC_NAMES_IO*
- *MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_ITEMS*
- *MSK_IPAR_WRITE_INT_CONSTRAINTS*
- *MSK_IPAR_WRITE_INT_HEAD*
- *MSK_IPAR_WRITE_INT_VARIABLES*
- *MSK_IPAR_WRITE_JSON_INDENTATION*
- *MSK_IPAR_WRITE_LP_FULL_OBJ*
- *MSK_IPAR_WRITE_LP_LINE_WIDTH*
- *MSK_IPAR_WRITE_MPS_FORMAT*
- *MSK_IPAR_WRITE_MPS_INT*
- *MSK_IPAR_WRITE_SOL_BARVARIABLES*
- *MSK_IPAR_WRITE_SOL_CONSTRAINTS*
- *MSK_IPAR_WRITE_SOL_HEAD*
- *MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES*
- *MSK_IPAR_WRITE_SOL_VARIABLES*
- *MSK_IPAR_WRITE_TASK_INC_SOL*
- *MSK_IPAR_WRITE_XML_MODE*

- *MSK_SPAR_BAS_SOL_FILE_NAME*
- *MSK_SPAR_DATA_FILE_NAME*
- *MSK_SPAR_DEBUG_FILE_NAME*
- *MSK_SPAR_INT_SOL_FILE_NAME*
- *MSK_SPAR_ITR_SOL_FILE_NAME*
- *MSK_SPAR_MIO_DEBUG_STRING*
- *MSK_SPAR_PARAM_COMMENT_SIGN*
- *MSK_SPAR_PARAM_READ_FILE_NAME*
- *MSK_SPAR_PARAM_WRITE_FILE_NAME*
- *MSK_SPAR_READ_MPS_BOU_NAME*
- *MSK_SPAR_READ_MPS_OBJ_NAME*
- *MSK_SPAR_READ_MPS_RAN_NAME*
- *MSK_SPAR_READ_MPS_RHS_NAME*
- *MSK_SPAR_SENSITIVITY_FILE_NAME*
- *MSK_SPAR_SENSITIVITY_RES_FILE_NAME*
- *MSK_SPAR_SOL_FILTER_XC_LOW*
- *MSK_SPAR_SOL_FILTER_XC_UPR*
- *MSK_SPAR_SOL_FILTER_XX_LOW*
- *MSK_SPAR_SOL_FILTER_XX_UPR*
- *MSK_SPAR_STAT_KEY*
- *MSK_SPAR_STAT_NAME*
- *MSK_SPAR_WRITE_LP_GEN_VAR_NAME*

Debugging

- *MSK_IPAR_AUTO_SORT_A_BEFORE_OPT*

Dual simplex

- *MSK_IPAR_SIM_DUAL_CRASH*
- *MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION*
- *MSK_IPAR_SIM_DUAL_SELECTION*

Infeasibility report

- *MSK_IPAR_INFEAS_GENERIC_NAMES*
- *MSK_IPAR_INFEAS_REPORT_LEVEL*
- *MSK_IPAR_LOG_INFEAS_ANA*

Interior-point method

- *MSK_DPAR_CHECK_CONVEXITY_REL_TOL*
- *MSK_DPAR_INTPNT_CO_TOL_DFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_INFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_MU_RED*
- *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_CO_TOL_PFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_QO_TOL_DFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_INFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_MU_RED*
- *MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_QO_TOL_PFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_TOL_DFEAS*
- *MSK_DPAR_INTPNT_TOL_DSAFE*
- *MSK_DPAR_INTPNT_TOL_INFEAS*
- *MSK_DPAR_INTPNT_TOL_MU_RED*
- *MSK_DPAR_INTPNT_TOL_PATH*
- *MSK_DPAR_INTPNT_TOL_PFEAS*
- *MSK_DPAR_INTPNT_TOL_PSAFE*
- *MSK_DPAR_INTPNT_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_TOL_REL_STEP*
- *MSK_DPAR_INTPNT_TOL_STEP_SIZE*
- *MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL*
- *MSK_IPAR_BI_IGNORE_MAX_ITER*
- *MSK_IPAR_BI_IGNORE_NUM_ERROR*
- *MSK_IPAR_INTPNT_BASIS*
- *MSK_IPAR_INTPNT_DIFF_STEP*
- *MSK_IPAR_INTPNT_HOTSTART*
- *MSK_IPAR_INTPNT_MAX_ITERATIONS*
- *MSK_IPAR_INTPNT_MAX_NUM_COR*
- *MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS*
- *MSK_IPAR_INTPNT_OFF_COL_TRH*
- *MSK_IPAR_INTPNT_ORDER_GP_NUM_SEEDS*
- *MSK_IPAR_INTPNT_ORDER_METHOD*

- *MSK_IPAR_INTPNT_PURIFY*
- *MSK_IPAR_INTPNT_REGULARIZATION_USE*
- *MSK_IPAR_INTPNT_SCALING*
- *MSK_IPAR_INTPNT_SOLVE_FORM*
- *MSK_IPAR_INTPNT_STARTING_POINT*
- *MSK_IPAR_LOG_INTPNT*

License manager

- *MSK_IPAR_CACHE_LICENSE*
- *MSK_IPAR_LICENSE_DEBUG*
- *MSK_IPAR_LICENSE_PAUSE_TIME*
- *MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS*
- *MSK_IPAR_LICENSE_TRH_EXPIRY_WRN*
- *MSK_IPAR_LICENSE_WAIT*

Logging

- *MSK_IPAR_LOG*
- *MSK_IPAR_LOG_ANA_PRO*
- *MSK_IPAR_LOG_BI*
- *MSK_IPAR_LOG_BI_FREQ*
- *MSK_IPAR_LOG_CUT_SECOND_OPT*
- *MSK_IPAR_LOG_EXPAND*
- *MSK_IPAR_LOG_FEAS_REPAIR*
- *MSK_IPAR_LOG_FILE*
- *MSK_IPAR_LOG_INCLUDE_SUMMARY*
- *MSK_IPAR_LOG_INFEAS_ANA*
- *MSK_IPAR_LOG_INTPNT*
- *MSK_IPAR_LOG_LOCAL_INFO*
- *MSK_IPAR_LOG_MIO*
- *MSK_IPAR_LOG_MIO_FREQ*
- *MSK_IPAR_LOG_ORDER*
- *MSK_IPAR_LOG_PRESOLVE*
- *MSK_IPAR_LOG_RESPONSE*
- *MSK_IPAR_LOG_SENSITIVITY*
- *MSK_IPAR_LOG_SENSITIVITY_OPT*
- *MSK_IPAR_LOG_SIM*
- *MSK_IPAR_LOG_SIM_FREQ*
- *MSK_IPAR_LOG_STORAGE*

Mixed-integer optimization

- *MSK_DPAR_MIO_DJC_MAX_BIGM*
- *MSK_DPAR_MIO_MAX_TIME*
- *MSK_DPAR_MIO_REL_GAP_CONST*
- *MSK_DPAR_MIO_TOL_ABS_GAP*
- *MSK_DPAR_MIO_TOL_ABS_RELAX_INT*
- *MSK_DPAR_MIO_TOL_FEAS*
- *MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT*
- *MSK_DPAR_MIO_TOL_REL_GAP*
- *MSK_IPAR_LOG_MIO*
- *MSK_IPAR_LOG_MIO_FREQ*
- *MSK_IPAR_MIO_BRANCH_DIR*
- *MSK_IPAR_MIO_CONIC_OUTER_APPROXIMATION*
- *MSK_IPAR_MIO_CONSTRUCT_SOL*
- *MSK_IPAR_MIO_CUT_CLIQUE*
- *MSK_IPAR_MIO_CUT_CMIR*
- *MSK_IPAR_MIO_CUT_GMI*
- *MSK_IPAR_MIO_CUT_IMPLIED_BOUND*
- *MSK_IPAR_MIO_CUT_KNAPSACK_COVER*
- *MSK_IPAR_MIO_CUT_LIPRO*
- *MSK_IPAR_MIO_CUT_SELECTION_LEVEL*
- *MSK_IPAR_MIO_DATA_PERMUTATION_METHOD*
- *MSK_IPAR_MIO_FEASPUMP_LEVEL*
- *MSK_IPAR_MIO_HEURISTIC_LEVEL*
- *MSK_IPAR_MIO_MAX_NUM_BRANCHES*
- *MSK_IPAR_MIO_MAX_NUM_RELAXS*
- *MSK_IPAR_MIO_MAX_NUM_ROOT_CUT_ROUNDS*
- *MSK_IPAR_MIO_MAX_NUM_SOLUTIONS*
- *MSK_IPAR_MIO_MEMORY_EMPHASIS_LEVEL*
- *MSK_IPAR_MIO_NODE_OPTIMIZER*
- *MSK_IPAR_MIO_NODE_SELECTION*
- *MSK_IPAR_MIO_NUMERICAL_EMPHASIS_LEVEL*
- *MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE*
- *MSK_IPAR_MIO_PROBING_LEVEL*
- *MSK_IPAR_MIO_PROPAGATE_OBJECTIVE_CONSTRAINT*
- *MSK_IPAR_MIO_QCQO_REFORMULATION_METHOD*

- *MSK_IPAR_MIO_RINS_MAX_NODES*
- *MSK_IPAR_MIO_ROOT_OPTIMIZER*
- *MSK_IPAR_MIO_ROOT_REPEAT_PREOLVE_LEVEL*
- *MSK_IPAR_MIO_SEED*
- *MSK_IPAR_MIO_SYMMETRY_LEVEL*
- *MSK_IPAR_MIO_VB_DETECTION_LEVEL*

Output information

- *MSK_IPAR_INFEAS_REPORT_LEVEL*
- *MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS*
- *MSK_IPAR_LICENSE_TRH_EXPIRY_WRN*
- *MSK_IPAR_LOG*
- *MSK_IPAR_LOG_BI*
- *MSK_IPAR_LOG_BI_FREQ*
- *MSK_IPAR_LOG_CUT_SECOND_OPT*
- *MSK_IPAR_LOG_EXPAND*
- *MSK_IPAR_LOG_FEAS_REPAIR*
- *MSK_IPAR_LOG_FILE*
- *MSK_IPAR_LOG_INCLUDE_SUMMARY*
- *MSK_IPAR_LOG_INFEAS_ANA*
- *MSK_IPAR_LOG_INTPNT*
- *MSK_IPAR_LOG_LOCAL_INFO*
- *MSK_IPAR_LOG_MIO*
- *MSK_IPAR_LOG_MIO_FREQ*
- *MSK_IPAR_LOG_ORDER*
- *MSK_IPAR_LOG_RESPONSE*
- *MSK_IPAR_LOG_SENSITIVITY*
- *MSK_IPAR_LOG_SENSITIVITY_OPT*
- *MSK_IPAR_LOG_SIM*
- *MSK_IPAR_LOG_SIM_FREQ*
- *MSK_IPAR_LOG_SIM_MINOR*
- *MSK_IPAR_LOG_STORAGE*
- *MSK_IPAR_MAX_NUM_WARNINGS*

Overall solver

- *MSK_IPAR_BI_CLEAN_OPTIMIZER*
- *MSK_IPAR_INFEAS_PREFER_PRIMAL*
- *MSK_IPAR_LICENSE_WAIT*
- *MSK_IPAR_MIO_MODE*
- *MSK_IPAR_OPTIMIZER*
- *MSK_IPAR_PRESOLVE_LEVEL*
- *MSK_IPAR_PRESOLVE_MAX_NUM_REDUCTIONS*
- *MSK_IPAR_PRESOLVE_USE*
- *MSK_IPAR_PRIMAL_REPAIR_OPTIMIZER*
- *MSK_IPAR_SENSITIVITY_ALL*
- *MSK_IPAR_SENSITIVITY_OPTIMIZER*
- *MSK_IPAR_SENSITIVITY_TYPE*
- *MSK_IPAR_SOLUTION_CALLBACK*

Overall system

- *MSK_IPAR_AUTO_UPDATE_SOL_INFO*
- *MSK_IPAR_LICENSE_WAIT*
- *MSK_IPAR_LOG_STORAGE*
- *MSK_IPAR_MT_SPINCOUNT*
- *MSK_IPAR_NUM_THREADS*
- *MSK_IPAR_REMOVE_UNUSED_SOLUTIONS*
- *MSK_IPAR_TIMING_LEVEL*
- *MSK_SPAR_REMOTE_OPTSERVER_HOST*
- *MSK_SPAR_REMOTE_TLS_CERT*
- *MSK_SPAR_REMOTE_TLS_CERT_PATH*

Presolve

- *MSK_DPAR_PRESOLVE_TOL_ABS_LINDEP*
- *MSK_DPAR_PRESOLVE_TOL_AIJ*
- *MSK_DPAR_PRESOLVE_TOL_PRIMAL_INFEAS_PERTURBATION*
- *MSK_DPAR_PRESOLVE_TOL_REL_LINDEP*
- *MSK_DPAR_PRESOLVE_TOL_S*
- *MSK_DPAR_PRESOLVE_TOL_X*
- *MSK_IPAR_MIO_PRESOLVE_AGGREGATOR_USE*
- *MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_FILL*

- *MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES*
- *MSK_IPAR_PRESOLVE_LEVEL*
- *MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH*
- *MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH*
- *MSK_IPAR_PRESOLVE_LINDEP_USE*
- *MSK_IPAR_PRESOLVE_MAX_NUM_PASS*
- *MSK_IPAR_PRESOLVE_MAX_NUM_REDUCTIONS*
- *MSK_IPAR_PRESOLVE_USE*

Primal simplex

- *MSK_IPAR_SIM_PRIMAL_CRASH*
- *MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION*
- *MSK_IPAR_SIM_PRIMAL_SELECTION*

Progress callback

- *MSK_IPAR_SOLUTION_CALLBACK*

Simplex optimizer

- *MSK_DPAR_BASIS_REL_TOL_S*
- *MSK_DPAR_BASIS_TOL_S*
- *MSK_DPAR_BASIS_TOL_X*
- *MSK_DPAR_SIM_LU_TOL_REL_PIV*
- *MSK_DPAR_SIMPLEX_ABS_TOL_PIV*
- *MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE*
- *MSK_IPAR_LOG_SIM*
- *MSK_IPAR_LOG_SIM_FREQ*
- *MSK_IPAR_LOG_SIM_MINOR*
- *MSK_IPAR_SENSITIVITY_OPTIMIZER*
- *MSK_IPAR_SIM_BASIS_FACTOR_USE*
- *MSK_IPAR_SIM_DEGEN*
- *MSK_IPAR_SIM_DETECT_PWL*
- *MSK_IPAR_SIM_DUAL_PHASEONE_METHOD*
- *MSK_IPAR_SIM_EXPLOIT_DUPVEC*
- *MSK_IPAR_SIM_HOTSTART*
- *MSK_IPAR_SIM_HOTSTART_LU*
- *MSK_IPAR_SIM_MAX_ITERATIONS*

- *MSK_IPAR_SIM_MAX_NUM_SETBACKS*
- *MSK_IPAR_SIM_NON_SINGULAR*
- *MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD*
- *MSK_IPAR_SIM_REFACTOR_FREQ*
- *MSK_IPAR_SIM_REFORMULATION*
- *MSK_IPAR_SIM_SAVE_LU*
- *MSK_IPAR_SIM_SCALING*
- *MSK_IPAR_SIM_SCALING_METHOD*
- *MSK_IPAR_SIM_SEED*
- *MSK_IPAR_SIM_SOLVE_FORM*
- *MSK_IPAR_SIM_STABILITY_PRIORITY*
- *MSK_IPAR_SIM_SWITCH_OPTIMIZER*

Solution input/output

- *MSK_IPAR_INFEAS_REPORT_AUTO*
- *MSK_IPAR_SOL_FILTER_KEEP_BASIC*
- *MSK_IPAR_SOL_FILTER_KEEP_RANGED*
- *MSK_IPAR_SOL_READ_NAME_WIDTH*
- *MSK_IPAR_SOL_READ_WIDTH*
- *MSK_IPAR_WRITE_BAS_CONSTRAINTS*
- *MSK_IPAR_WRITE_BAS_HEAD*
- *MSK_IPAR_WRITE_BAS_VARIABLES*
- *MSK_IPAR_WRITE_INT_CONSTRAINTS*
- *MSK_IPAR_WRITE_INT_HEAD*
- *MSK_IPAR_WRITE_INT_VARIABLES*
- *MSK_IPAR_WRITE_SOL_BARVARIABLES*
- *MSK_IPAR_WRITE_SOL_CONSTRAINTS*
- *MSK_IPAR_WRITE_SOL_HEAD*
- *MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES*
- *MSK_IPAR_WRITE_SOL_VARIABLES*
- *MSK_SPAR_BAS_SOL_FILE_NAME*
- *MSK_SPAR_INT_SOL_FILE_NAME*
- *MSK_SPAR_ITR_SOL_FILE_NAME*
- *MSK_SPAR_SOL_FILTER_XC_LOW*
- *MSK_SPAR_SOL_FILTER_XC_UPR*
- *MSK_SPAR_SOL_FILTER_XX_LOW*
- *MSK_SPAR_SOL_FILTER_XX_UPR*

Termination criteria

- *MSK_DPAR_BASIS_REL_TOL_S*
- *MSK_DPAR_BASIS_TOL_S*
- *MSK_DPAR_BASIS_TOL_X*
- *MSK_DPAR_INTPNT_CO_TOL_DFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_INFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_MU_RED*
- *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_CO_TOL_PFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_QO_TOL_DFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_INFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_MU_RED*
- *MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_QO_TOL_PFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_TOL_DFEAS*
- *MSK_DPAR_INTPNT_TOL_INFEAS*
- *MSK_DPAR_INTPNT_TOL_MU_RED*
- *MSK_DPAR_INTPNT_TOL_PFEAS*
- *MSK_DPAR_INTPNT_TOL_REL_GAP*
- *MSK_DPAR_LOWER_OBJ_CUT*
- *MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH*
- *MSK_DPAR_MIO_MAX_TIME*
- *MSK_DPAR_MIO_REL_GAP_CONST*
- *MSK_DPAR_MIO_TOL_REL_GAP*
- *MSK_DPAR_OPTIMIZER_MAX_TIME*
- *MSK_DPAR_UPPER_OBJ_CUT*
- *MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH*
- *MSK_IPAR_BI_MAX_ITERATIONS*
- *MSK_IPAR_INTPNT_MAX_ITERATIONS*
- *MSK_IPAR_MIO_MAX_NUM_BRANCHES*
- *MSK_IPAR_MIO_MAX_NUM_ROOT_CUT_ROUNDS*
- *MSK_IPAR_MIO_MAX_NUM_SOLUTIONS*
- *MSK_IPAR_SIM_MAX_ITERATIONS*

Other

- *MSK_IPAR_COMPRESS_STATFILE*
- *MSK_IPAR_NG*
- *MSK_IPAR_REMOTE_USE_COMPRESSION*

11.2 Parameters (alphabetical list sorted by type)

- *Double parameters*
- *Integer parameters*
- *String parameters*

11.2.1 Double parameters

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.

Default 1e-6

Accepted [0.0; +inf]

Example `mosek -d MSK_DPAR_ANA_SOL_INFEAS_TOL 1e-6 file`

Groups *Analysis*

MSK_DPAR_BASIS_REL_TOL_S

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

Default 1.0e-12

Accepted [0.0; +inf]

Example `mosek -d MSK_DPAR_BASIS_REL_TOL_S 1.0e-12 file`

Groups *Simplex optimizer, Termination criteria*

MSK_DPAR_BASIS_TOL_S

Maximum absolute dual bound violation in an optimal basic solution.

Default 1.0e-6

Accepted [1.0e-9; +inf]

Example `mosek -d MSK_DPAR_BASIS_TOL_S 1.0e-6 file`

Groups *Simplex optimizer, Termination criteria*

MSK_DPAR_BASIS_TOL_X

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

Default 1.0e-6

Accepted [1.0e-9; +inf]

Example `mosek -d MSK_DPAR_BASIS_TOL_X 1.0e-6 file`

Groups *Simplex optimizer, Termination criteria*

MSK_DPAR_CHECK_CONVEXITY_REL_TOL

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

A problem is declared non-convex if negative (positive) pivot elements are detected in the Cholesky factor of a matrix which is required to be PSD (NSD). This parameter 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:

$$d_i \leq -|Q_{ii}| \text{check_convexity_rel_tol}$$

Default 1e-10
Accepted [0; +inf]
Example mosek -d MSK_DPAR_CHECK_CONVEXITY_REL_TOL 1e-10 file
Groups *Interior-point method*

MSK_DPAR_DATA_SYM_MAT_TOL

Absolute zero tolerance for elements in symmetric matrices. If any value in a symmetric matrix is smaller than this parameter in absolute terms **MOSEK** will treat the values as zero and generate a warning.

Default 1.0e-12
Accepted [1.0e-16; 1.0e-6]
Example mosek -d MSK_DPAR_DATA_SYM_MAT_TOL 1.0e-12 file
Groups *Data check*

MSK_DPAR_DATA_SYM_MAT_TOL_HUGE

An element in a symmetric matrix which is larger than this value in absolute size causes an error.

Default 1.0e20
Accepted [0.0; +inf]
Example mosek -d MSK_DPAR_DATA_SYM_MAT_TOL_HUGE 1.0e20 file
Groups *Data check*

MSK_DPAR_DATA_SYM_MAT_TOL_LARGE

An element in a symmetric matrix which is larger than this value in absolute size causes a warning message to be printed.

Default 1.0e10
Accepted [0.0; +inf]
Example mosek -d MSK_DPAR_DATA_SYM_MAT_TOL_LARGE 1.0e10 file
Groups *Data check*

MSK_DPAR_DATA_TOL_AIJ_HUGE

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

Default 1.0e20
Accepted [0.0; +inf]
Example mosek -d MSK_DPAR_DATA_TOL_AIJ_HUGE 1.0e20 file
Groups *Data check*

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.

Default 1.0e10
Accepted [0.0; +inf]
Example mosek -d MSK_DPAR_DATA_TOL_AIJ_LARGE 1.0e10 file
Groups *Data check*

MSK_DPAR_DATA_TOL_BOUND_INF

Any bound which in absolute value is greater than this parameter is considered infinite.

Default 1.0e16
Accepted [0.0; +inf]
Example mosek -d MSK_DPAR_DATA_TOL_BOUND_INF 1.0e16 file
Groups *Data check*

MSK_DPAR_DATA_TOL_BOUND_WRN

If a bound value is larger than this value in absolute size, then a warning message is issued.

Default 1.0e8
Accepted [0.0; +inf]
Example mosek -d MSK_DPAR_DATA_TOL_BOUND_WRN 1.0e8 file
Groups *Data check*

MSK_DPAR_DATA_TOL_C_HUGE

An element in c which is larger than the value of this parameter in absolute terms is considered to be huge and generates an error.

Default 1.0e16
Accepted [0.0; +inf]
Example mosek -d MSK_DPAR_DATA_TOL_C_HUGE 1.0e16 file
Groups *Data check*

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.

Default 1.0e8
Accepted [0.0; +inf]
Example mosek -d MSK_DPAR_DATA_TOL_CJ_LARGE 1.0e8 file
Groups *Data check*

MSK_DPAR_DATA_TOL_QIJ

Absolute zero tolerance for elements in Q matrices.

Default 1.0e-16
Accepted [0.0; +inf]
Example mosek -d MSK_DPAR_DATA_TOL_QIJ 1.0e-16 file
Groups *Data check*

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 upper bound is considered identical.

Default 1.0e-8
Accepted [0.0; +inf]
Example mosek -d MSK_DPAR_DATA_TOL_X 1.0e-8 file
Groups *Data check*

MSK_DPAR_INTPNT_CO_TOL_DFEAS

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

Default 1.0e-8
Accepted [0.0; 1.0]
Example mosek -d MSK_DPAR_INTPNT_CO_TOL_DFEAS 1.0e-8 file
See also *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*
Groups *Interior-point method, Termination criteria, Conic interior-point method*

MSK_DPAR_INTPNT_CO_TOL_INFEAS

Infeasibility tolerance used by the interior-point optimizer for conic problems. Controls when the interior-point optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

Default 1.0e-12
Accepted [0.0; 1.0]
Example mosek -d MSK_DPAR_INTPNT_CO_TOL_INFEAS 1.0e-12 file
Groups *Interior-point method, Termination criteria, Conic interior-point method*

MSK_DPAR_INTPNT_CO_TOL_MU_RED

Relative complementarity gap tolerance used by the interior-point optimizer for conic problems.

Default 1.0e-8

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_CO_TOL_MU_RED 1.0e-8 file`

Groups *Interior-point method, Termination criteria, Conic interior-point method*

MSK_DPAR_INTPNT_CO_TOL_NEAR_REL

Optimality tolerance used by the interior-point optimizer for conic problems. If **MOSEK** cannot compute a solution that has the prescribed accuracy then it will check if the solution found satisfies the termination criteria with all tolerances multiplied by the value of this parameter. If yes, then the solution is also declared optimal.

Default 1000

Accepted [1.0; +inf]

Example `mosek -d MSK_DPAR_INTPNT_CO_TOL_NEAR_REL 1000 file`

Groups *Interior-point method, Termination criteria, Conic interior-point method*

MSK_DPAR_INTPNT_CO_TOL_PFEAS

Primal feasibility tolerance used by the interior-point optimizer for conic problems.

Default 1.0e-8

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_CO_TOL_PFEAS 1.0e-8 file`

See also [*MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*](#)

Groups *Interior-point method, Termination criteria, Conic interior-point method*

MSK_DPAR_INTPNT_CO_TOL_REL_GAP

Relative gap termination tolerance used by the interior-point optimizer for conic problems.

Default 1.0e-8

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_CO_TOL_REL_GAP 1.0e-8 file`

See also [*MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*](#)

Groups *Interior-point method, Termination criteria, Conic interior-point method*

MSK_DPAR_INTPNT_QO_TOL_DFEAS

Dual feasibility tolerance used by the interior-point optimizer for quadratic problems.

Default 1.0e-8

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_QO_TOL_DFEAS 1.0e-8 file`

See also [*MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*](#)

Groups *Interior-point method, Termination criteria*

MSK_DPAR_INTPNT_QO_TOL_INFEAS

Infeasibility tolerance used by the interior-point optimizer for quadratic problems. Controls when the interior-point optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

Default 1.0e-12

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_QO_TOL_INFEAS 1.0e-12 file`

Groups *Interior-point method, Termination criteria*

MSK_DPAR_INTPNT_QO_TOL_MU_RED

Relative complementarity gap tolerance used by the interior-point optimizer for quadratic problems.

Default 1.0e-8

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_QO_TOL_MU_RED 1.0e-8 file`

Groups *Interior-point method, Termination criteria*

MSK_DPAR_INTPNT_QO_TOL_NEAR_REL

Optimality tolerance used by the interior-point optimizer for quadratic problems. If **MOSEK** cannot compute a solution that has the prescribed accuracy then it will check if the solution found satisfies the termination criteria with all tolerances multiplied by the value of this parameter. If yes, then the solution is also declared optimal.

Default 1000

Accepted [1.0; +inf]

Example `mosek -d MSK_DPAR_INTPNT_QO_TOL_NEAR_REL 1000 file`

Groups *Interior-point method, Termination criteria*

MSK_DPAR_INTPNT_QO_TOL_PFEAS

Primal feasibility tolerance used by the interior-point optimizer for quadratic problems.

Default 1.0e-8

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_QO_TOL_PFEAS 1.0e-8 file`

See also [*MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*](#)

Groups *Interior-point method, Termination criteria*

MSK_DPAR_INTPNT_QO_TOL_REL_GAP

Relative gap termination tolerance used by the interior-point optimizer for quadratic problems.

Default 1.0e-8

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_QO_TOL_REL_GAP 1.0e-8 file`

See also [*MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*](#)

Groups *Interior-point method, Termination criteria*

MSK_DPAR_INTPNT_TOL_DFEAS

Dual feasibility tolerance used by the interior-point optimizer for linear problems.

Default 1.0e-8

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_TOL_DFEAS 1.0e-8 file`

Groups *Interior-point method, Termination criteria*

MSK_DPAR_INTPNT_TOL_DSAFE

Controls the initial dual starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly and/or the constraint or variable bounds are very large, then it might be worthwhile to increase this value.

Default 1.0

Accepted [1.0e-4; +inf]

Example `mosek -d MSK_DPAR_INTPNT_TOL_DSAFE 1.0 file`

Groups *Interior-point method*

MSK_DPAR_INTPNT_TOL_INFEAS

Infeasibility tolerance used by the interior-point optimizer for linear problems. Controls when the interior-point optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

Default 1.0e-10

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_TOL_INFEAS 1.0e-10 file`

Groups *Interior-point method, Termination criteria*

MSK_DPAR_INTPNT_TOL_MU_RED

Relative complementarity gap tolerance used by the interior-point optimizer for linear problems.

Default 1.0e-16

Accepted [0.0; 1.0]

Example mosek -d MSK_DPAR_INTPNT_TOL_MU_RED 1.0e-16 file

Groups *Interior-point method, Termination criteria*

MSK_DPAR_INTPNT_TOL_PATH

Controls how close the interior-point optimizer follows the central path. A large value of this parameter means the central path is followed very closely. On numerically unstable problems it may be worthwhile to increase this parameter.

Default 1.0e-8

Accepted [0.0; 0.9999]

Example mosek -d MSK_DPAR_INTPNT_TOL_PATH 1.0e-8 file

Groups *Interior-point method*

MSK_DPAR_INTPNT_TOL_PFEAS

Primal feasibility tolerance used by the interior-point optimizer for linear problems.

Default 1.0e-8

Accepted [0.0; 1.0]

Example mosek -d MSK_DPAR_INTPNT_TOL_PFEAS 1.0e-8 file

Groups *Interior-point method, Termination criteria*

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.

Default 1.0

Accepted [1.0e-4; +inf]

Example mosek -d MSK_DPAR_INTPNT_TOL_PSAFE 1.0 file

Groups *Interior-point method*

MSK_DPAR_INTPNT_TOL_REL_GAP

Relative gap termination tolerance used by the interior-point optimizer for linear problems.

Default 1.0e-8

Accepted [1.0e-14; +inf]

Example mosek -d MSK_DPAR_INTPNT_TOL_REL_GAP 1.0e-8 file

Groups *Termination criteria, Interior-point method*

MSK_DPAR_INTPNT_TOL_REL_STEP

Relative step size to the boundary for linear and quadratic optimization problems.

Default 0.9999

Accepted [1.0e-4; 0.999999]

Example mosek -d MSK_DPAR_INTPNT_TOL_REL_STEP 0.9999 file

Groups *Interior-point method*

MSK_DPAR_INTPNT_TOL_STEP_SIZE

Minimal step size tolerance. If the step size falls below the value of this parameter, then the interior-point optimizer assumes that it is stalled. In other words the interior-point optimizer does not make any progress and therefore it is better to stop.

Default 1.0e-6

Accepted [0.0; 1.0]

Example mosek -d MSK_DPAR_INTPNT_TOL_STEP_SIZE 1.0e-6 file

Groups *Interior-point method*

MSK_DPAR_LOWER_OBJ_CUT

If either a primal or dual feasible solution is found proving that the optimal objective value is outside the interval [*MSK_DPAR_LOWER_OBJ_CUT*, *MSK_DPAR_UPPER_OBJ_CUT*], then **MOSEK** is terminated.

Default -1.0e30

Accepted [-inf; +inf]

Example mosek -d MSK_DPAR_LOWER_OBJ_CUT -1.0e30 file

See also *MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH*

Groups *Termination criteria*

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 $-\infty$.

Default -0.5e30

Accepted [-inf; +inf]

Example mosek -d MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH -0.5e30 file

Groups *Termination criteria*

MSK_DPAR_MIO_DJC_MAX_BIGM

Maximum allowed big-M value when reformulating disjunctive constraints to linear constraints. Higher values make it more likely that a disjunction is reformulated to linear constraints, but also increase the risk of numerical problems.

Default 1.0e6

Accepted [0; +inf]

Example mosek -d MSK_DPAR_MIO_DJC_MAX_BIGM 1.0e6 file

Groups *Mixed-integer optimization*

MSK_DPAR_MIO_MAX_TIME

This parameter limits the maximum time spent by the mixed-integer optimizer. A negative number means infinity.

Default -1.0

Accepted [-inf; +inf]

Example mosek -d MSK_DPAR_MIO_MAX_TIME -1.0 file

Groups *Mixed-integer optimization, Termination criteria*

MSK_DPAR_MIO_REL_GAP_CONST

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

Default 1.0e-10

Accepted [1.0e-15; +inf]

Example mosek -d MSK_DPAR_MIO_REL_GAP_CONST 1.0e-10 file

Groups *Mixed-integer optimization, Termination criteria*

MSK_DPAR_MIO_TOL_ABS_GAP

Absolute optimality tolerance employed by the mixed-integer optimizer.

Default 0.0

Accepted [0.0; +inf]

Example mosek -d MSK_DPAR_MIO_TOL_ABS_GAP 0.0 file

Groups *Mixed-integer optimization*

MSK_DPAR_MIO_TOL_ABS_RELAX_INT

Absolute integer feasibility tolerance. If the distance to the nearest integer is less than this tolerance then an integer constraint is assumed to be satisfied.

Default 1.0e-5

Accepted [1e-9; +inf]

Example mosek -d MSK_DPAR_MIO_TOL_ABS_RELAX_INT 1.0e-5 file

Groups *Mixed-integer optimization*

MSK_DPAR_MIO_TOL_FEAS

Feasibility tolerance for mixed integer solver.

Default 1.0e-6

Accepted [1e-9; 1e-3]

Example mosek -d MSK_DPAR_MIO_TOL_FEAS 1.0e-6 file

Groups *Mixed-integer optimization*

MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT

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

Default 0.0

Accepted [0.0; 1.0]

Example mosek -d MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT 0.0 file

Groups *Mixed-integer optimization*

MSK_DPAR_MIO_TOL_REL_GAP

Relative optimality tolerance employed by the mixed-integer optimizer.

Default 1.0e-4

Accepted [0.0; +inf]

Example mosek -d MSK_DPAR_MIO_TOL_REL_GAP 1.0e-4 file

Groups *Mixed-integer optimization, Termination criteria*

MSK_DPAR_OPTIMIZER_MAX_TIME

Maximum amount of time the optimizer is allowed to spent on the optimization. A negative number means infinity.

Default -1.0

Accepted [-inf; +inf]

Example mosek -d MSK_DPAR_OPTIMIZER_MAX_TIME -1.0 file

Groups *Termination criteria*

MSK_DPAR_PREOLVE_TOL_ABS_LINDEP

Absolute tolerance employed by the linear dependency checker.

Default 1.0e-6

Accepted [0.0; +inf]

Example mosek -d MSK_DPAR_PREOLVE_TOL_ABS_LINDEP 1.0e-6 file

Groups *Presolve*

MSK_DPAR_PREOLVE_TOL_AIJ

Absolute zero tolerance employed for a_{ij} in the presolve.

Default 1.0e-12

Accepted [1.0e-15; +inf]

Example mosek -d MSK_DPAR_PREOLVE_TOL_AIJ 1.0e-12 file

Groups *Presolve*

MSK_DPAR_PREOLVE_TOL_PRIMAL_INFEAS_PERTURBATION

The presolve is allowed to perturb a bound on a constraint or variable by this amount if it removes an infeasibility.

Default 1.0e-6

Accepted [0.0; +inf]

Example mosek -d MSK_DPAR_PREOLVE_TOL_PRIMAL_INFEAS_PERTURBATION 1.0e-6 file

Groups *Presolve*

MSK_DPAR_PREOLVE_TOL_REL_LINDEP

Relative tolerance employed by the linear dependency checker.

Default 1.0e-10

Accepted [0.0; +inf]

Example mosek -d MSK_DPAR_PREOLVE_TOL_REL_LINDEP 1.0e-10 file

Groups *Presolve*

MSK_DPAR_PREOLVE_TOL_S

Absolute zero tolerance employed for s_i in the presolve.

Default 1.0e-8

Accepted [0.0; +inf]

Example mosek -d MSK_DPAR_PREOLVE_TOL_S 1.0e-8 file

Groups *Presolve*

MSK_DPAR_PREOLVE_TOL_X

Absolute zero tolerance employed for x_j in the presolve.

Default 1.0e-8

Accepted [0.0; +inf]

Example mosek -d MSK_DPAR_PREOLVE_TOL_X 1.0e-8 file

Groups *Presolve*

MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL

This parameter determines when columns are dropped in incomplete Cholesky factorization during reformulation of quadratic problems.

Default 1e-15

Accepted [0; +inf]

Example mosek -d MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL 1e-15 file

Groups *Interior-point method*

MSK_DPAR_SEMIDEFINITE_TOL_APPROX

Tolerance to define a matrix to be positive semidefinite.

Default 1.0e-10

Accepted [1.0e-15; +inf]

Example mosek -d MSK_DPAR_SEMIDEFINITE_TOL_APPROX 1.0e-10 file

Groups *Data check*

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.

Default 0.01

Accepted [1.0e-6; 0.999999]

Example mosek -d MSK_DPAR_SIM_LU_TOL_REL_PIV 0.01 file

Groups *Basis identification, Simplex optimizer*

MSK_DPAR_SIMPLEX_ABS_TOL_PIV

Absolute pivot tolerance employed by the simplex optimizers.

Default 1.0e-7

Accepted [1.0e-12; +inf]

Example mosek -d MSK_DPAR_SIMPLEX_ABS_TOL_PIV 1.0e-7 file

Groups *Simplex optimizer*

MSK_DPAR_UPPER_OBJ_CUT

If either a primal or dual feasible solution is found proving that the optimal objective value is outside the interval [*MSK_DPAR_LOWER_OBJ_CUT*, *MSK_DPAR_UPPER_OBJ_CUT*], then **MOSEK** is terminated.

Default 1.0e30

Accepted [-inf; +inf]

Example mosek -d MSK_DPAR_UPPER_OBJ_CUT 1.0e30 file

See also *MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH*

Groups *Termination criteria*

MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH

If the upper objective cut is greater than the value of this parameter, then the upper objective cut *MSK_DPAR_UPPER_OBJ_CUT* is treated as ∞ .

Default 0.5e30

Accepted [-inf; +inf]

Example mosek -d MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH 0.5e30 file

Groups *Termination criteria*

11.2.2 Integer parameters

MSK_IPAR_ANA_SOL_BASIS

Controls whether the basis matrix is analyzed in solution analyzer.

Default *ON*

Accepted *ON*, *OFF*

Example mosek -d MSK_IPAR_ANA_SOL_BASIS MSK_ON file

Groups *Analysis*

MSK_IPAR_ANA_SOL_PRINT_VIOLATED

A parameter of the problem analyzer. Controls whether a list of violated constraints is printed. All constraints violated by more than the value set by the parameter *MSK_DPAR_ANA_SOL_INFEAS_TOL* will be printed.

Default *OFF*

Accepted *ON*, *OFF*

Example mosek -d MSK_IPAR_ANA_SOL_PRINT_VIOLATED MSK_OFF file

Groups *Analysis*

MSK_IPAR_AUTO_SORT_A_BEFORE_OPT

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.

Default *OFF*

Accepted *ON*, *OFF*

Example mosek -d MSK_IPAR_AUTO_SORT_A_BEFORE_OPT MSK_OFF file

Groups *Debugging*

MSK_IPAR_AUTO_UPDATE_SOL_INFO

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

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_AUTO_UPDATE_SOL_INFO MSK_OFF file`

Groups *Overall system*

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.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE MSK_OFF file`

Groups *Simplex optimizer*

MSK_IPAR_BI_CLEAN_OPTIMIZER

Controls which simplex optimizer is used in the clean-up phase. Anything else than *MSK_OPTIMIZER_PRIMAL_SIMPLEX* or *MSK_OPTIMIZER_DUAL_SIMPLEX* is equivalent to *MSK_OPTIMIZER_FREE_SIMPLEX*.

Default *FREE*

Accepted *FREE, INTPNT, CONIC, PRIMAL_SIMPLEX, DUAL_SIMPLEX, FREE_SIMPLEX, MIXED_INT*

Example `mosek -d MSK_IPAR_BI_CLEAN_OPTIMIZER MSK_OPTIMIZER_FREE file`

Groups *Basis identification, Overall solver*

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*.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_BI_IGNORE_MAX_ITER MSK_OFF file`

Groups *Interior-point method, Basis identification*

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*.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_BI_IGNORE_NUM_ERROR MSK_OFF file`

Groups *Interior-point method, Basis identification*

MSK_IPAR_BI_MAX_ITERATIONS

Controls the maximum number of simplex iterations allowed to optimize a basis after the basis identification.

Default 1000000

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_BI_MAX_ITERATIONS 1000000 file`

Groups *Basis identification, Termination criteria*

MSK_IPAR_CACHE_LICENSE

Specifies if the license is kept checked out for the lifetime of the **MOSEK** environment/model/process (*MSK_ON*) or returned to the server immediately after the optimization (*MSK_OFF*).

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

Default *ON*
Accepted *ON, OFF*
Example `mosek -d MSK_IPAR_CACHE_LICENSE MSK_ON file`
Groups *License manager*

MSK_IPAR_CHECK_CONVEXITY
Specify the level of convexity check on quadratic problems.

Default *FULL*
Accepted *NONE, SIMPLE, FULL*
Example `mosek -d MSK_IPAR_CHECK_CONVEXITY MSK_CHECK_CONVEXITY_FULL file`
Groups *Data check*

MSK_IPAR_COMPRESS_STATFILE
Control compression of stat files.

Default *ON*
Accepted *ON, OFF*
Example `mosek -d MSK_IPAR_COMPRESS_STATFILE MSK_ON file`

MSK_IPAR_INFEAS_GENERIC_NAMES
Controls whether generic names are used when an infeasible subproblem is created.

Default *OFF*
Accepted *ON, OFF*
Example `mosek -d MSK_IPAR_INFEAS_GENERIC_NAMES MSK_OFF file`
Groups *Infeasibility report*

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.

Default *ON*
Accepted *ON, OFF*
Example `mosek -d MSK_IPAR_INFEAS_PREFER_PRIMAL MSK_ON file`
Groups *Overall solver*

MSK_IPAR_INFEAS_REPORT_AUTO
Controls whether an infeasibility report is automatically produced after the optimization if the problem is primal or dual infeasible.

Default *OFF*
Accepted *ON, OFF*
Example `mosek -d MSK_IPAR_INFEAS_REPORT_AUTO MSK_OFF file`
Groups *Data input/output, Solution input/output*

MSK_IPAR_INFEAS_REPORT_LEVEL
Controls the amount of information presented in an infeasibility report. Higher values imply more information.

Default *1*
Accepted *[0; +inf]*
Example `mosek -d MSK_IPAR_INFEAS_REPORT_LEVEL 1 file`
Groups *Infeasibility report, Output information*

MSK_IPAR_INTPNT_BASIS
Controls whether the interior-point optimizer also computes an optimal basis.

Default *ALWAYS*
Accepted *NEVER, ALWAYS, NO_ERROR, IF_FEASIBLE, RESERVED*

Example `mosek -d MSK_IPAR_INTPNT_BASIS MSK_BI_ALWAYS file`

See also `MSK_IPAR_BI_IGNORE_MAX_ITER`, `MSK_IPAR_BI_IGNORE_NUM_ERROR`,
`MSK_IPAR_BI_MAX_ITERATIONS`, `MSK_IPAR_BI_CLEAN_OPTIMIZER`

Groups *Interior-point method, Basis identification*

MSK_IPAR_INTPNT_DIFF_STEP

Controls whether different step sizes are allowed in the primal and dual space.

Default *ON*

Accepted

- *ON*: Different step sizes are allowed.
- *OFF*: Different step sizes are not allowed.

Example `mosek -d MSK_IPAR_INTPNT_DIFF_STEP MSK_ON file`

Groups *Interior-point method*

MSK_IPAR_INTPNT_HOTSTART

Currently not in use.

Default *NONE*

Accepted *NONE, PRIMAL, DUAL, PRIMAL_DUAL*

Example `mosek -d MSK_IPAR_INTPNT_HOTSTART MSK_INTPNT_HOTSTART_NONE file`

Groups *Interior-point method*

MSK_IPAR_INTPNT_MAX_ITERATIONS

Controls the maximum number of iterations allowed in the interior-point optimizer.

Default 400

Accepted $[0; +\infty]$

Example `mosek -d MSK_IPAR_INTPNT_MAX_ITERATIONS 400 file`

Groups *Interior-point method, Termination criteria*

MSK_IPAR_INTPNT_MAX_NUM_COR

Controls the maximum number of correctors allowed by the multiple corrector procedure. A negative value means that **MOSEK** is making the choice.

Default -1

Accepted $[-1; +\infty]$

Example `mosek -d MSK_IPAR_INTPNT_MAX_NUM_COR -1 file`

Groups *Interior-point method*

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.

Default -1

Accepted $[-\infty; +\infty]$

Example `mosek -d MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS -1 file`

Groups *Interior-point method*

MSK_IPAR_INTPNT_OFF_COL_TRH

Controls how many offending columns are detected in the Jacobian of the constraint matrix.

0	no detection
1	aggressive detection
> 1	higher values mean less aggressive detection

Default 40

Accepted $[0; +\infty]$

Example `mosek -d MSK_IPAR_INTPNT_OFF_COL_TRH 40 file`

Groups *Interior-point method*

MSK_IPAR_INTPNT_ORDER_GP_NUM_SEEDS

The GP ordering is dependent on a random seed. Therefore, trying several random seeds may lead to a better ordering. This parameter controls the number of random seeds tried.

A value of 0 means that MOSEK makes the choice.

Default 0

Accepted [0; +inf]

Example mosek -d MSK_IPAR_INTPNT_ORDER_GP_NUM_SEEDS 0 file

Groups *Interior-point method*

MSK_IPAR_INTPNT_ORDER_METHOD

Controls the ordering strategy used by the interior-point optimizer when factorizing the Newton equation system.

Default *FREE*

Accepted *FREE, APPMINLOC, EXPERIMENTAL, TRY_GRAPHPAR, FORCE_GRAPHPAR, NONE*

Example mosek -d MSK_IPAR_INTPNT_ORDER_METHOD MSK_ORDER_METHOD_FREE
file

Groups *Interior-point method*

MSK_IPAR_INTPNT_PURIFY

Currently not in use.

Default *NONE*

Accepted *NONE, PRIMAL, DUAL, PRIMAL_DUAL, AUTO*

Example mosek -d MSK_IPAR_INTPNT_PURIFY MSK_PURIFY_NONE file

Groups *Interior-point method*

MSK_IPAR_INTPNT_REGULARIZATION_USE

Controls whether regularization is allowed.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_INTPNT_REGULARIZATION_USE MSK_ON file

Groups *Interior-point method*

MSK_IPAR_INTPNT_SCALING

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

Default *FREE*

Accepted *FREE, NONE*

Example mosek -d MSK_IPAR_INTPNT_SCALING MSK_SCALING_FREE file

Groups *Interior-point method*

MSK_IPAR_INTPNT_SOLVE_FORM

Controls whether the primal or the dual problem is solved.

Default *FREE*

Accepted *FREE, PRIMAL, DUAL*

Example mosek -d MSK_IPAR_INTPNT_SOLVE_FORM MSK_SOLVE_FREE file

Groups *Interior-point method*

MSK_IPAR_INTPNT_STARTING_POINT

Starting point used by the interior-point optimizer.

Default *FREE*

Accepted *FREE, GUESS, CONSTANT, SATISFY_BOUNDS*

Example mosek -d MSK_IPAR_INTPNT_STARTING_POINT MSK_STARTING_POINT_FREE
file

Groups *Interior-point method*

MSK_IPAR_LICENSE_DEBUG

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

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_LICENSE_DEBUG MSK_OFF file

Groups *License manager*

MSK_IPAR_LICENSE_PAUSE_TIME

If *MSK_IPAR_LICENSE_WAIT* is *MSK_ON* and no license is available, then **MOSEK** sleeps a number of milliseconds between each check of whether a license has become free.

Default 100

Accepted [0; 1000000]

Example mosek -d MSK_IPAR_LICENSE_PAUSE_TIME 100 file

Groups *License manager*

MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS

Controls whether license features expire warnings are suppressed.

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS MSK_OFF file

Groups *License manager, Output information*

MSK_IPAR_LICENSE_TRH_EXPIRY_WRN

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

Default 7

Accepted [0; +inf]

Example mosek -d MSK_IPAR_LICENSE_TRH_EXPIRY_WRN 7 file

Groups *License manager, Output information*

MSK_IPAR_LICENSE_WAIT

If all licenses are in use **MOSEK** returns with an error code. However, by turning on this parameter **MOSEK** will wait for an available license.

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_LICENSE_WAIT MSK_OFF file

Groups *Overall solver, Overall system, License manager*

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.

Default 10

Accepted [0; +inf]

Example mosek -d MSK_IPAR_LOG 10 file

See also *MSK_IPAR_LOG_CUT_SECOND_OPT*

Groups *Output information, Logging*

MSK_IPAR_LOG_ANA_PRO

Controls amount of output from the problem analyzer.

Default 1

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_ANA_PRO 1 file`

Groups *Analysis, Logging*

MSK_IPAR_LOG_BI

Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.

Default 1

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_BI 1 file`

Groups *Basis identification, Output information, Logging*

MSK_IPAR_LOG_BI_FREQ

Controls how frequently the optimizer outputs information about the basis identification and how frequent the user-defined callback function is called.

Default 2500

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_BI_FREQ 2500 file`

Groups *Basis identification, Output information, Logging*

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.

Default 0

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_CHECK_CONVEXITY 0 file`

Groups *Data check*

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.

Default 1

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_CUT_SECOND_OPT 1 file`

See also *MSK_IPAR_LOG*, *MSK_IPAR_LOG_INTPNT*, *MSK_IPAR_LOG_MIO*,
MSK_IPAR_LOG_SIM

Groups *Output information, Logging*

MSK_IPAR_LOG_EXPAND

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

Default 0

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_EXPAND 0 file`

Groups *Output information, Logging*

MSK_IPAR_LOG_FEAS_REPAIR

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

Default 1

Accepted [0; +inf]

Example mosek -d MSK_IPAR_LOG_FEAS_REPAIR 1 file

Groups *Output information, Logging*

MSK_IPAR_LOG_FILE

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

Default 1

Accepted [0; +inf]

Example mosek -d MSK_IPAR_LOG_FILE 1 file

Groups *Data input/output, Output information, Logging*

MSK_IPAR_LOG_INCLUDE_SUMMARY

Not relevant for this API.

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_LOG_INCLUDE_SUMMARY MSK_OFF file

Groups *Output information, Logging*

MSK_IPAR_LOG_INFEAS_ANA

Controls amount of output printed by the infeasibility analyzer procedures. A higher level implies that more information is logged.

Default 1

Accepted [0; +inf]

Example mosek -d MSK_IPAR_LOG_INFEAS_ANA 1 file

Groups *Infeasibility report, Output information, Logging*

MSK_IPAR_LOG_INTPNT

Controls amount of output printed by the interior-point optimizer. A higher level implies that more information is logged.

Default 1

Accepted [0; +inf]

Example mosek -d MSK_IPAR_LOG_INTPNT 1 file

Groups *Interior-point method, Output information, Logging*

MSK_IPAR_LOG_LOCAL_INFO

Controls whether local identifying information like environment variables, filenames, IP addresses etc. are printed to the log.

Note that this will only affect some functions. Some functions that specifically emit system information will not be affected.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_LOG_LOCAL_INFO MSK_ON file

Groups *Output information, Logging*

MSK_IPAR_LOG_MIO

Controls the log level for the mixed-integer optimizer. A higher level implies that more information is logged.

Default 4

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_MIO 4 file`

Groups *Mixed-integer optimization, Output information, Logging*

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.

Default 10

Accepted [-inf; +inf]

Example `mosek -d MSK_IPAR_LOG_MIO_FREQ 10 file`

Groups *Mixed-integer optimization, Output information, Logging*

MSK_IPAR_LOG_ORDER

If turned on, then factor lines are added to the log.

Default 1

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_ORDER 1 file`

Groups *Output information, Logging*

MSK_IPAR_LOG_PRESOLVE

Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.

Default 1

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_PRESOLVE 1 file`

Groups *Logging*

MSK_IPAR_LOG_RESPONSE

Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.

Default 0

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_RESPONSE 0 file`

Groups *Output information, Logging*

MSK_IPAR_LOG_SENSITIVITY

Controls the amount of logging during the sensitivity analysis.

- 0. Means no logging information is produced.
- 1. Timing information is printed.
- 2. Sensitivity results are printed.

Default 1

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_SENSITIVITY 1 file`

Groups *Output information, Logging*

MSK_IPAR_LOG_SENSITIVITY_OPT

Controls the amount of logging from the optimizers employed during the sensitivity analysis. 0 means no logging information is produced.

Default 0

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_SENSITIVITY_OPT 0 file`

Groups *Output information, Logging*

MSK_IPAR_LOG_SIM

Controls amount of output printed by the simplex optimizer. A higher level implies that more information is logged.

Default 4

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_SIM 4 file`

Groups *Simplex optimizer, Output information, Logging*

MSK_IPAR_LOG_SIM_FREQ

Controls how frequent the simplex optimizer outputs information about the optimization and how frequent the user-defined callback function is called.

Default 1000

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_SIM_FREQ 1000 file`

Groups *Simplex optimizer, Output information, Logging*

MSK_IPAR_LOG_SIM_MINOR

Currently not in use.

Default 1

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_SIM_MINOR 1 file`

Groups *Simplex optimizer, Output information*

MSK_IPAR_LOG_STORAGE

When turned on, **MOSEK** prints messages regarding the storage usage and allocation.

Default 0

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_STORAGE 0 file`

Groups *Output information, Overall system, Logging*

MSK_IPAR_MAX_NUM_WARNINGS

Each warning is shown a limited number of times controlled by this parameter. A negative value is identical to infinite number of times.

Default 10

Accepted [-inf; +inf]

Example `mosek -d MSK_IPAR_MAX_NUM_WARNINGS 10 file`

Groups *Output information*

MSK_IPAR_MIO_BRANCH_DIR

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

Default *FREE*

Accepted *FREE, UP, DOWN, NEAR, FAR, ROOT_LP, GUIDED, PSEUDOCOST*

Example `mosek -d MSK_IPAR_MIO_BRANCH_DIR MSK_BRANCH_DIR_FREE file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CONIC_OUTER_APPROXIMATION

If this option is turned on outer approximation is used when solving relaxations of conic problems; otherwise interior point is used.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_MIO_CONIC_OUTER_APPROXIMATION MSK_OFF file`

Groups *Mixed-integer optimization*

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.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_MIO_CONSTRUCT_SOL MSK_OFF file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_CLIQUE

Controls whether clique cuts should be generated.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_MIO_CUT_CLIQUE MSK_ON file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_CMIR

Controls whether mixed integer rounding cuts should be generated.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_MIO_CUT_CMIR MSK_ON file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_GMI

Controls whether GMI cuts should be generated.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_MIO_CUT_GMI MSK_ON file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_IMPLIED_BOUND

Controls whether implied bound cuts should be generated.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_MIO_CUT_IMPLIED_BOUND MSK_ON file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_KNAPSACK_COVER

Controls whether knapsack cover cuts should be generated.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_MIO_CUT_KNAPSACK_COVER MSK_OFF file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_LIPRO

Controls whether lift-and-project cuts should be generated.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_MIO_CUT_LIPRO MSK_OFF file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_SELECTION_LEVEL

Controls how aggressively generated cuts are selected to be included in the relaxation.

- -1. The optimizer chooses the level of cut selection
- 0. Generated cuts less likely to be added to the relaxation
- 1. Cuts are more aggressively selected to be included in the relaxation

Default -1

Accepted [-1; +1]

Example `mosek -d MSK_IPAR_MIO_CUT_SELECTION_LEVEL -1 file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_DATA_PERMUTATION_METHOD

Controls what problem data permutation method is applied to mixed-integer problems.

Default *NONE*

Accepted *NONE, CYCLIC_SHIFT, RANDOM*

Example `mosek -d MSK_IPAR_MIO_DATA_PERMUTATION_METHOD`

`MSK_MIO_DATA_PERMUTATION_METHOD_NONE file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_FEASPUMP_LEVEL

Controls the way the Feasibility Pump heuristic is employed by the mixed-integer optimizer.

- -1. The optimizer chooses how the Feasibility Pump is used
- 0. The Feasibility Pump is disabled
- 1. The Feasibility Pump is enabled with an effort to improve solution quality
- 2. The Feasibility Pump is enabled with an effort to reach feasibility early

Default -1

Accepted [-1; 2]

Example `mosek -d MSK_IPAR_MIO_FEASPUMP_LEVEL -1 file`

Groups *Mixed-integer optimization*

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.

Default -1

Accepted [-inf; +inf]

Example `mosek -d MSK_IPAR_MIO_HEURISTIC_LEVEL -1 file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_MAX_NUM_BRANCHES

Maximum number of branches allowed during the branch and bound search. A negative value means infinite.

Default -1

Accepted [-inf; +inf]

Example `mosek -d MSK_IPAR_MIO_MAX_NUM_BRANCHES -1 file`

Groups *Mixed-integer optimization, Termination criteria*

MSK_IPAR_MIO_MAX_NUM_RELAXS

Maximum number of relaxations allowed during the branch and bound search. A negative value means infinite.

Default -1

Accepted [-inf; +inf]

Example mosek -d MSK_IPAR_MIO_MAX_NUM_RELAXS -1 file

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_MAX_NUM_ROOT_CUT_ROUNDS

Maximum number of cut separation rounds at the root node.

Default 100

Accepted [0; +inf]

Example mosek -d MSK_IPAR_MIO_MAX_NUM_ROOT_CUT_ROUNDS 100 file

Groups *Mixed-integer optimization, Termination criteria*

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 $n > 0$, then the mixed-integer optimizer will be terminated when n feasible solutions have been located.

Default -1

Accepted [-inf; +inf]

Example mosek -d MSK_IPAR_MIO_MAX_NUM_SOLUTIONS -1 file

Groups *Mixed-integer optimization, Termination criteria*

MSK_IPAR_MIO_MEMORY_EMPHASIS_LEVEL

Controls how much emphasis is put on reducing memory usage. Being more conservative about memory usage may come at the cost of decreased solution speed.

- 0. The optimizer chooses
- 1. More emphasis is put on reducing memory usage and less on speed

Default 0

Accepted [0; +1]

Example mosek -d MSK_IPAR_MIO_MEMORY_EMPHASIS_LEVEL 0 file

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_MODE

Controls whether the optimizer includes the integer restrictions and disjunctive constraints when solving a (mixed) integer optimization problem.

Default *SATISFIED*

Accepted *IGNORED, SATISFIED*

Example mosek -d MSK_IPAR_MIO_MODE MSK_MIO_MODE_SATISFIED file

Groups *Overall solver*

MSK_IPAR_MIO_NODE_OPTIMIZER

Controls which optimizer is employed at the non-root nodes in the mixed-integer optimizer.

Default *FREE*

Accepted *FREE, INTPNT, CONIC, PRIMAL_SIMPLEX, DUAL_SIMPLEX, FREE_SIMPLEX, MIXED_INT*

Example mosek -d MSK_IPAR_MIO_NODE_OPTIMIZER MSK_OPTIMIZER_FREE file

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_NODE_SELECTION

Controls the node selection strategy employed by the mixed-integer optimizer.

Default *FREE*

Accepted *FREE, FIRST, BEST, PSEUDO*

Example mosek -d MSK_IPAR_MIO_NODE_SELECTION MSK_MIO_NODE_SELECTION_FREE file

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_NUMERICAL_EMPHASIS_LEVEL

Controls how much emphasis is put on reducing numerical problems possibly at the expense of solution speed.

- 0. The optimizer chooses
- 1. More emphasis is put on reducing numerical problems
- 2. Even more emphasis

Default 0

Accepted [0; +2]

Example mosek -d MSK_IPAR_MIO_NUMERICAL_EMPHASIS_LEVEL 0 file

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE

Enables or disables perspective reformulation in presolve.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE MSK_ON file

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_PRESOLVE_AGGREGATOR_USE

Controls if the aggregator should be used.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_MIO_PRESOLVE_AGGREGATOR_USE MSK_ON file

Groups *Presolve*

MSK_IPAR_MIO_PROBING_LEVEL

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

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

Default -1

Accepted [-1; 3]

Example mosek -d MSK_IPAR_MIO_PROBING_LEVEL -1 file

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_PROPAGATE_OBJECTIVE_CONSTRAINT

Use objective domain propagation.

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_MIO_PROPAGATE_OBJECTIVE_CONSTRAINT MSK_OFF
file

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_QCQO_REFORMULATION_METHOD

Controls what reformulation method is applied to mixed-integer quadratic problems.

Default *FREE*

Accepted *FREE, NONE, LINEARIZATION, EIGEN_VAL_METHOD, DIAG_SDP, RELAX_SDP*

Example mosek -d MSK_IPAR_MIO_QCQO_REFORMULATION_METHOD
MSK_MIO_QCQO_REFORMULATION_METHOD_FREE file

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_RINS_MAX_NODES

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

Default -1

Accepted [-1; +inf]

Example `mosek -d MSK_IPAR_MIO_RINS_MAX_NODES -1 file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_ROOT_OPTIMIZER

Controls which optimizer is employed at the root node in the mixed-integer optimizer.

Default *FREE*

Accepted *FREE*, *INTPNT*, *CONIC*, *PRIMAL_SIMPLEX*, *DUAL_SIMPLEX*, *FREE_SIMPLEX*, *MIXED_INT*

Example `mosek -d MSK_IPAR_MIO_ROOT_OPTIMIZER MSK_OPTIMIZER_FREE file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_ROOT_REPEAT_PRESOLVE_LEVEL

Controls whether presolve can be repeated at root node.

- -1. The optimizer chooses whether presolve is repeated
- 0. Never repeat presolve
- 1. Always repeat presolve

Default -1

Accepted [-1; 1]

Example `mosek -d MSK_IPAR_MIO_ROOT_REPEAT_PRESOLVE_LEVEL -1 file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_SEED

Sets the random seed used for randomization in the mixed integer optimizer. Selecting a different seed can change the path the optimizer takes to the optimal solution.

Default 42

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_MIO_SEED 42 file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_SYMMETRY_LEVEL

Controls the amount of symmetry detection and handling employed by the mixed-integer optimizer in presolve.

- -1. The optimizer chooses the level of symmetry detection and handling employed
- 0. Symmetry detection and handling is disabled
- 1. A low amount of symmetry detection and handling is employed
- 2. A medium amount of symmetry detection and handling is employed
- 3. A high amount of symmetry detection and handling is employed
- 4. An extremely high amount of symmetry detection and handling is employed

Default -1

Accepted [-1; 4]

Example `mosek -d MSK_IPAR_MIO_SYMMETRY_LEVEL -1 file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_VB_DETECTION_LEVEL

Controls how much effort is put into detecting variable bounds.

- -1. The optimizer chooses
- 0. No variable bounds are detected
- 1. Only detect variable bounds that are directly represented in the problem
- 2. Detect variable bounds in probing

Default -1

Accepted [-1; +2]

Example `mosek -d MSK_IPAR_MIO_VB_DETECTION_LEVEL -1 file`

Groups *Mixed-integer optimization*

MSK_IPAR_MT_SPINCOUNT

Set the number of iterations to spin before sleeping.

Default 0

Accepted [0; 1000000000]

Example `mosek -d MSK_IPAR_MT_SPINCOUNT 0 file`

Groups *Overall system*

MSK_IPAR_NG

Not in use.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_NG MSK_OFF file`

MSK_IPAR_NUM_THREADS

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

Default 0

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_NUM_THREADS 0 file`

Groups *Overall system*

MSK_IPAR_OPF_WRITE_HEADER

Write a text header with date and **MOSEK** version in an OPF file.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_OPF_WRITE_HEADER MSK_ON file`

Groups *Data input/output*

MSK_IPAR_OPF_WRITE_HINTS

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

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_OPF_WRITE_HINTS MSK_ON file`

Groups *Data input/output*

MSK_IPAR_OPF_WRITE_LINE_LENGTH

Aim to keep lines in OPF files not much longer than this.

Default 80

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_OPF_WRITE_LINE_LENGTH 80 file`

Groups *Data input/output*

MSK_IPAR_OPF_WRITE_PARAMETERS

Write a parameter section in an OPF file.

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_OPF_WRITE_PARAMETERS MSK_OFF file

Groups *Data input/output*

MSK_IPAR_OPF_WRITE_PROBLEM

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

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_OPF_WRITE_PROBLEM MSK_ON file

Groups *Data input/output*

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.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_OPF_WRITE_SOL_BAS MSK_ON file

Groups *Data input/output*

MSK_IPAR_OPF_WRITE_SOL_ITG

If *MSK_IPAR_OPF_WRITE_SOLUTIONS* is *MSK_ON* and an integer solution is defined, write the integer solution in OPF files.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_OPF_WRITE_SOL_ITG MSK_ON file

Groups *Data input/output*

MSK_IPAR_OPF_WRITE_SOL_ITR

If *MSK_IPAR_OPF_WRITE_SOLUTIONS* is *MSK_ON* and an interior solution is defined, write the interior solution in OPF files.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_OPF_WRITE_SOL_ITR MSK_ON file

Groups *Data input/output*

MSK_IPAR_OPF_WRITE_SOLUTIONS

Enable inclusion of solutions in the OPF files.

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_OPF_WRITE_SOLUTIONS MSK_OFF file

Groups *Data input/output*

MSK_IPAR_OPTIMIZER

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

Default *FREE*

Accepted *FREE, INTPNT, CONIC, PRIMAL_SIMPLEX, DUAL_SIMPLEX, FREE_SIMPLEX, MIXED_INT*

Example mosek -d MSK_IPAR_OPTIMIZER MSK_OPTIMIZER_FREE file

Groups *Overall solver*

MSK_IPAR_PARAM_READ_CASE_NAME

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

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_PARAM_READ_CASE_NAME MSK_ON file`

Groups *Data input/output*

MSK_IPAR_PARAM_READ_IGN_ERROR

If turned on, then errors in parameter settings is ignored.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_PARAM_READ_IGN_ERROR MSK_OFF file`

Groups *Data input/output*

MSK_IPAR_PREOLVE_ELIMINATOR_MAX_FILL

Controls the maximum amount of fill-in that can be created by one pivot in the elimination phase of the presolve. A negative value means the parameter value is selected automatically.

Default *-1*

Accepted *[-inf; +inf]*

Example `mosek -d MSK_IPAR_PREOLVE_ELIMINATOR_MAX_FILL -1 file`

Groups *Presolve*

MSK_IPAR_PREOLVE_ELIMINATOR_MAX_NUM_TRIES

Control the maximum number of times the eliminator is tried. A negative value implies **MOSEK** decides.

Default *-1*

Accepted *[-inf; +inf]*

Example `mosek -d MSK_IPAR_PREOLVE_ELIMINATOR_MAX_NUM_TRIES -1 file`

Groups *Presolve*

MSK_IPAR_PREOLVE_LEVEL

Currently not used.

Default *-1*

Accepted *[-inf; +inf]*

Example `mosek -d MSK_IPAR_PREOLVE_LEVEL -1 file`

Groups *Overall solver, Presolve*

MSK_IPAR_PREOLVE_LINDEP_ABS_WORK_TRH

Controls linear dependency check in presolve. The linear dependency check is potentially computationally expensive.

Default *100*

Accepted *[-inf; +inf]*

Example `mosek -d MSK_IPAR_PREOLVE_LINDEP_ABS_WORK_TRH 100 file`

Groups *Presolve*

MSK_IPAR_PREOLVE_LINDEP_REL_WORK_TRH

Controls linear dependency check in presolve. The linear dependency check is potentially computationally expensive.

Default *100*

Accepted *[-inf; +inf]*

Example `mosek -d MSK_IPAR_PREOLVE_LINDEP_REL_WORK_TRH 100 file`

Groups *Presolve*

MSK_IPAR_PRESOLVE_LINDEP_USE

Controls whether the linear constraints are checked for linear dependencies.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_PRESOLVE_LINDEP_USE MSK_ON file`

Groups *Presolve*

MSK_IPAR_PRESOLVE_MAX_NUM_PASS

Control the maximum number of times presolve passes over the problem. A negative value implies **MOSEK** decides.

Default *-1*

Accepted *[-inf; +inf]*

Example `mosek -d MSK_IPAR_PRESOLVE_MAX_NUM_PASS -1 file`

Groups *Presolve*

MSK_IPAR_PRESOLVE_MAX_NUM_REDUCTIONS

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

Default *-1*

Accepted *[-inf; +inf]*

Example `mosek -d MSK_IPAR_PRESOLVE_MAX_NUM_REDUCTIONS -1 file`

Groups *Overall solver, Presolve*

MSK_IPAR_PRESOLVE_USE

Controls whether the presolve is applied to a problem before it is optimized.

Default *FREE*

Accepted *OFF, ON, FREE*

Example `mosek -d MSK_IPAR_PRESOLVE_USE MSK_PRESOLVE_MODE_FREE file`

Groups *Overall solver, Presolve*

MSK_IPAR_PRIMAL_REPAIR_OPTIMIZER

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

Default *FREE*

Accepted *FREE, INTPNT, CONIC, PRIMAL_SIMPLEX, DUAL_SIMPLEX, FREE_SIMPLEX, MIXED_INT*

Example `mosek -d MSK_IPAR_PRIMAL_REPAIR_OPTIMIZER MSK_OPTIMIZER_FREE file`

Groups *Overall solver*

MSK_IPAR_PTF_WRITE_PARAMETERS

If *MSK_IPAR_PTF_WRITE_PARAMETERS* is *MSK_ON*, the parameters section is written.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_PTF_WRITE_PARAMETERS MSK_OFF file`

Groups *Data input/output*

MSK_IPAR_PTF_WRITE_SOLUTIONS

If *MSK_IPAR_PTF_WRITE_SOLUTIONS* is *MSK_ON*, the solution section is written if any solutions are available, otherwise solution section is not written even if solutions are available.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_PTF_WRITE_SOLUTIONS MSK_OFF file`

Groups *Data input/output*

MSK_IPAR_PTF_WRITE_TRANSFORM

If *MSK_IPAR_PTF_WRITE_TRANSFORM* is *MSK_ON*, constraint blocks with identifiable conic slacks are transformed into conic constraints and the slacks are eliminated.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_PTF_WRITE_TRANSFORM MSK_ON file`

Groups *Data input/output*

MSK_IPAR_READ_DEBUG

Turns on additional debugging information when reading files.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_READ_DEBUG MSK_OFF file`

Groups *Data input/output*

MSK_IPAR_READ_KEEP_FREE_CON

Controls whether the free constraints are included in the problem.

Default *OFF*

Accepted

- *ON*: The free constraints are kept.
- *OFF*: The free constraints are discarded.

Example `mosek -d MSK_IPAR_READ_KEEP_FREE_CON MSK_OFF file`

Groups *Data input/output*

MSK_IPAR_READ_MPS_FORMAT

Controls how strictly the MPS file reader interprets the MPS format.

Default *FREE*

Accepted *STRICT, RELAXED, FREE, CPLEX*

Example `mosek -d MSK_IPAR_READ_MPS_FORMAT MSK_MPS_FORMAT_FREE file`

Groups *Data input/output*

MSK_IPAR_READ_MPS_WIDTH

Controls the maximal number of characters allowed in one line of the MPS file.

Default 1024

Accepted [80; +inf]

Example `mosek -d MSK_IPAR_READ_MPS_WIDTH 1024 file`

Groups *Data input/output*

MSK_IPAR_READ_TASK_IGNORE_PARAM

Controls whether **MOSEK** should ignore the parameter setting defined in the task file and use the default parameter setting instead.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_READ_TASK_IGNORE_PARAM MSK_OFF file`

Groups *Data input/output*

MSK_IPAR_REMOTE_USE_COMPRESSION

Use compression when sending data to an optimization server.

Default *ZSTD*

Accepted *NONE, FREE, GZIP, ZSTD*

Example `mosek -d MSK_IPAR_REMOTE_USE_COMPRESSION MSK_COMPRESS_ZSTD file`

MSK_IPAR_REMOVE_UNUSED_SOLUTIONS

Removes unused solutions before the optimization is performed.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_REMOVE_UNUSED_SOLUTIONS MSK_OFF file`

Groups *Overall system*

MSK_IPAR_SENSITIVITY_ALL

Not applicable.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_SENSITIVITY_ALL MSK_OFF file`

Groups *Overall solver*

MSK_IPAR_SENSITIVITY_OPTIMIZER

Controls which optimizer is used for optimal partition sensitivity analysis.

Default *FREE_SIMPLEX*

Accepted *FREE, INTPNT, CONIC, PRIMAL_SIMPLEX, DUAL_SIMPLEX, FREE_SIMPLEX, MIXED_INT*

Example `mosek -d MSK_IPAR_SENSITIVITY_OPTIMIZER MSK_OPTIMIZER_FREE_SIMPLEX file`

Groups *Overall solver, Simplex optimizer*

MSK_IPAR_SENSITIVITY_TYPE

Controls which type of sensitivity analysis is to be performed.

Default *BASIS*

Accepted *BASIS*

Example `mosek -d MSK_IPAR_SENSITIVITY_TYPE MSK_SENSITIVITY_TYPE_BASIS file`

Groups *Overall solver*

MSK_IPAR_SIM_BASIS_FACTOR_USE

Controls whether an 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.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_SIM_BASIS_FACTOR_USE MSK_ON file`

Groups *Simplex optimizer*

MSK_IPAR_SIM_DEGEN

Controls how aggressively degeneration is handled.

Default *FREE*

Accepted *NONE, FREE, AGGRESSIVE, MODERATE, MINIMUM*

Example `mosek -d MSK_IPAR_SIM_DEGEN MSK_SIM_DEGEN_FREE file`

Groups *Simplex optimizer*

MSK_IPAR_SIM_DETECT_PWL

Not in use.

Default *ON*

Accepted

- *ON*: PWL are detected.
- *OFF*: PWL are not detected.

Example `mosek -d MSK_IPAR_SIM_DETECT_PWL MSK_ON file`

Groups *Simplex optimizer*

MSK_IPAR_SIM_DUAL_CRASH

Controls whether crashing is performed in the dual simplex optimizer. If this parameter is set to x , then a crash will be performed if a basis consists of more than $(100 - x) \bmod f_v$ entries, where f_v is the number of fixed variables.

Default 90

Accepted $[0; +\infty]$

Example `mosek -d MSK_IPAR_SIM_DUAL_CRASH 90 file`

Groups *Dual simplex*

MSK_IPAR_SIM_DUAL_PHASEONE_METHOD

An experimental feature.

Default 0

Accepted $[0; 10]$

Example `mosek -d MSK_IPAR_SIM_DUAL_PHASEONE_METHOD 0 file`

Groups *Simplex optimizer*

MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION

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 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.

Default 50

Accepted $[0; 100]$

Example `mosek -d MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION 50 file`

Groups *Dual simplex*

MSK_IPAR_SIM_DUAL_SELECTION

Controls the choice of the incoming variable, known as the selection strategy, in the dual simplex optimizer.

Default *FREE*

Accepted *FREE, FULL, ASE, DEVEX, SE, PARTIAL*

Example `mosek -d MSK_IPAR_SIM_DUAL_SELECTION MSK_SIM_SELECTION_FREE file`

Groups *Dual simplex*

MSK_IPAR_SIM_EXPLOIT_DUPVEC

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

Default *OFF*

Accepted *ON, OFF, FREE*

Example `mosek -d MSK_IPAR_SIM_EXPLOIT_DUPVEC MSK_SIM_EXPLOIT_DUPVEC_OFF file`

Groups *Simplex optimizer*

MSK_IPAR_SIM_HOTSTART

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

Default *FREE*

Accepted *NONE, FREE, STATUS_KEYS*

Example `mosek -d MSK_IPAR_SIM_HOTSTART MSK_SIM_HOTSTART_FREE file`

Groups *Simplex optimizer*

MSK_IPAR_SIM_HOTSTART_LU

Determines if the simplex optimizer should exploit the initial factorization.

Default *ON*

Accepted

- *ON*: Factorization is reused if possible.
- *OFF*: Factorization is recomputed.

Example `mosek -d MSK_IPAR_SIM_HOTSTART_LU MSK_ON file`

Groups *Simplex optimizer*

MSK_IPAR_SIM_MAX_ITERATIONS

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

Default 10000000

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_SIM_MAX_ITERATIONS 10000000 file`

Groups *Simplex optimizer, Termination criteria*

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.

Default 250

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_SIM_MAX_NUM_SETBACKS 250 file`

Groups *Simplex optimizer*

MSK_IPAR_SIM_NON_SINGULAR

Controls if the simplex optimizer ensures a non-singular basis, if possible.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_SIM_NON_SINGULAR MSK_ON file`

Groups *Simplex optimizer*

MSK_IPAR_SIM_PRIMAL_CRASH

Controls whether crashing is performed in the primal simplex optimizer. In general, if a basis consists of more than (100-this parameter value)% fixed variables, then a crash will be performed.

Default 90

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_SIM_PRIMAL_CRASH 90 file`

Groups *Primal simplex*

MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD

An experimental feature.

Default 0

Accepted [0; 10]

Example `mosek -d MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD 0 file`

Groups *Simplex optimizer*

MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION

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 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.

Default 50

Accepted [0; 100]

Example mosek -d MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION 50 file

Groups *Primal simplex*

MSK_IPAR_SIM_PRIMAL_SELECTION

Controls the choice of the incoming variable, known as the selection strategy, in the primal simplex optimizer.

Default *FREE*

Accepted *FREE, FULL, ASE, DEVEX, SE, PARTIAL*

Example mosek -d MSK_IPAR_SIM_PRIMAL_SELECTION MSK_SIM_SELECTION_FREE
file

Groups *Primal simplex*

MSK_IPAR_SIM_REFACTOR_FREQ

Controls how frequent the basis is refactorized. The value 0 means that the optimizer determines the best point of refactorization. It is strongly recommended NOT to change this parameter.

Default 0

Accepted [0; +inf]

Example mosek -d MSK_IPAR_SIM_REFACTOR_FREQ 0 file

Groups *Simplex optimizer*

MSK_IPAR_SIM_REFORMULATION

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

Default *OFF*

Accepted *ON, OFF, FREE, AGGRESSIVE*

Example mosek -d MSK_IPAR_SIM_REFORMULATION MSK_SIM_REFORMULATION_OFF
file

Groups *Simplex optimizer*

MSK_IPAR_SIM_SAVE_LU

Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_SIM_SAVE_LU MSK_OFF file

Groups *Simplex optimizer*

MSK_IPAR_SIM_SCALING

Controls how much effort is used in scaling the problem before a simplex optimizer is used.

Default *FREE*

Accepted *FREE, NONE*

Example mosek -d MSK_IPAR_SIM_SCALING MSK_SCALING_FREE file

Groups *Simplex optimizer*

MSK_IPAR_SIM_SCALING_METHOD

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

Default *POW2*

Accepted *POW2, FREE*

Example mosek -d MSK_IPAR_SIM_SCALING_METHOD MSK_SCALING_METHOD_POW2
file

Groups *Simplex optimizer*

MSK_IPAR_SIM_SEED

Sets the random seed used for randomization in the simplex optimizers.

Default 23456

Accepted [0; 32749]

Example mosek -d MSK_IPAR_SIM_SEED 23456 file

Groups *Simplex optimizer*

MSK_IPAR_SIM_SOLVE_FORM

Controls whether the primal or the dual problem is solved by the primal-/dual-simplex optimizer.

Default *FREE*

Accepted *FREE, PRIMAL, DUAL*

Example mosek -d MSK_IPAR_SIM_SOLVE_FORM MSK_SOLVE_FREE file

Groups *Simplex optimizer*

MSK_IPAR_SIM_STABILITY_PRIORITY

Controls how high priority the numerical stability should be given.

Default 50

Accepted [0; 100]

Example mosek -d MSK_IPAR_SIM_STABILITY_PRIORITY 50 file

Groups *Simplex optimizer*

MSK_IPAR_SIM_SWITCH_OPTIMIZER

The simplex optimizer sometimes chooses to solve the dual problem instead of the primal problem. This implies that if you have chosen to use the dual simplex optimizer and the problem is dualized, then it actually makes sense to use the primal simplex optimizer instead. If this parameter is on and the problem is dualized and furthermore the simplex optimizer is chosen to be the primal (dual) one, then it is switched to the dual (primal).

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_SIM_SWITCH_OPTIMIZER MSK_OFF file

Groups *Simplex optimizer*

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.

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_SOL_FILTER_KEEP_BASIC MSK_OFF file

Groups *Solution input/output*

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.

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_SOL_FILTER_KEEP_RANGED MSK_OFF file

Groups *Solution input/output*

MSK_IPAR_SOL_READ_NAME_WIDTH

When a solution is read by **MOSEK** and some constraint, variable or cone names contain blanks, then a maximum name width must be specified. A negative value implies that no name contains blanks.

Default -1

Accepted [-inf; +inf]

Example mosek -d MSK_IPAR_SOL_READ_NAME_WIDTH -1 file

Groups *Data input/output, Solution input/output*

MSK_IPAR_SOL_READ_WIDTH

Controls the maximal acceptable width of line in the solutions when read by **MOSEK**.

Default 1024

Accepted [80; +inf]

Example mosek -d MSK_IPAR_SOL_READ_WIDTH 1024 file

Groups *Data input/output, Solution input/output*

MSK_IPAR_SOLUTION_CALLBACK

Indicates whether solution callbacks will be performed during the optimization.

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_SOLUTION_CALLBACK MSK_OFF file

Groups *Progress callback, Overall solver*

MSK_IPAR_TIMING_LEVEL

Controls the amount of timing performed inside **MOSEK**.

Default 1

Accepted [0; +inf]

Example mosek -d MSK_IPAR_TIMING_LEVEL 1 file

Groups *Overall system*

MSK_IPAR_WRITE_BAS_CONSTRAINTS

Controls whether the constraint section is written to the basic solution file.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_WRITE_BAS_CONSTRAINTS MSK_ON file

Groups *Data input/output, Solution input/output*

MSK_IPAR_WRITE_BAS_HEAD

Controls whether the header section is written to the basic solution file.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_WRITE_BAS_HEAD MSK_ON file

Groups *Data input/output, Solution input/output*

MSK_IPAR_WRITE_BAS_VARIABLES

Controls whether the variables section is written to the basic solution file.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_WRITE_BAS_VARIABLES MSK_ON file

Groups *Data input/output, Solution input/output*

MSK_IPAR_WRITE_COMPRESSION

Controls whether the data file is compressed while it is written. 0 means no compression while higher values mean more compression.

Default 9

Accepted [0; +inf]

Example mosek -d MSK_IPAR_WRITE_COMPRESSION 9 file

Groups *Data input/output*

MSK_IPAR_WRITE_DATA_PARAM

If this option is turned on the parameter settings are written to the data file as parameters.

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_WRITE_DATA_PARAM MSK_OFF file

Groups *Data input/output*

MSK_IPAR_WRITE_FREE_CON

Controls whether the free constraints are written to the data file.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_WRITE_FREE_CON MSK_ON file

Groups *Data input/output*

MSK_IPAR_WRITE_GENERIC_NAMES

Controls whether generic names should be used instead of user-defined names when writing to the data file.

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_WRITE_GENERIC_NAMES MSK_OFF file

Groups *Data input/output*

MSK_IPAR_WRITE_GENERIC_NAMES_IO

Index origin used in generic names.

Default 1

Accepted [0; +inf]

Example mosek -d MSK_IPAR_WRITE_GENERIC_NAMES_IO 1 file

Groups *Data input/output*

MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_ITEMS

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

Default *OFF*

Accepted

- *ON*: Ignore items that cannot be written to the current output file format.
- *OFF*: Produce an error if the problem contains items that cannot be written to the current output file format.

Example mosek -d MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_ITEMS MSK_OFF file

Groups *Data input/output*

MSK_IPAR_WRITE_INT_CONSTRAINTS

Controls whether the constraint section is written to the integer solution file.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_WRITE_INT_CONSTRAINTS MSK_ON file

Groups *Data input/output, Solution input/output*

MSK_IPAR_WRITE_INT_HEAD

Controls whether the header section is written to the integer solution file.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_WRITE_INT_HEAD MSK_ON file`

Groups *Data input/output, Solution input/output*

MSK_IPAR_WRITE_INT_VARIABLES

Controls whether the variables section is written to the integer solution file.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_WRITE_INT_VARIABLES MSK_ON file`

Groups *Data input/output, Solution input/output*

MSK_IPAR_WRITE_JSON_INDENTATION

When set, the JSON task and solution files are written with indentation for better readability.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_WRITE_JSON_INDENTATION MSK_OFF file`

Groups *Data input/output*

MSK_IPAR_WRITE_LP_FULL_OBJ

Write all variables, including the ones with 0-coefficients, in the objective.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_WRITE_LP_FULL_OBJ MSK_ON file`

Groups *Data input/output*

MSK_IPAR_WRITE_LP_LINE_WIDTH

Maximum width of line in an LP file written by **MOSEK**.

Default 80

Accepted [40; +inf]

Example `mosek -d MSK_IPAR_WRITE_LP_LINE_WIDTH 80 file`

Groups *Data input/output*

MSK_IPAR_WRITE_MPS_FORMAT

Controls in which format the MPS is written.

Default *FREE*

Accepted *STRICT, RELAXED, FREE, CPLEX*

Example `mosek -d MSK_IPAR_WRITE_MPS_FORMAT MSK_MPS_FORMAT_FREE file`

Groups *Data input/output*

MSK_IPAR_WRITE_MPS_INT

Controls if marker records are written to the MPS file to indicate whether variables are integer restricted.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_WRITE_MPS_INT MSK_ON file`

Groups *Data input/output*

MSK_IPAR_WRITE_SOL_BARVARIABLES

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

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_WRITE_SOL_BARVARIABLES MSK_ON file

Groups *Data input/output, Solution input/output*

MSK_IPAR_WRITE_SOL_CONSTRAINTS

Controls whether the constraint section is written to the solution file.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_WRITE_SOL_CONSTRAINTS MSK_ON file

Groups *Data input/output, Solution input/output*

MSK_IPAR_WRITE_SOL_HEAD

Controls whether the header section is written to the solution file.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_WRITE_SOL_HEAD MSK_ON file

Groups *Data input/output, Solution input/output*

MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES

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

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES MSK_OFF file

Groups *Data input/output, Solution input/output*

MSK_IPAR_WRITE_SOL_VARIABLES

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

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_WRITE_SOL_VARIABLES MSK_ON file

Groups *Data input/output, Solution input/output*

MSK_IPAR_WRITE_TASK_INC_SOL

Controls whether the solutions are stored in the task file too.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_WRITE_TASK_INC_SOL MSK_ON file

Groups *Data input/output*

MSK_IPAR_WRITE_XML_MODE

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

Default *ROW*

Accepted *ROW, COL*

Example mosek -d MSK_IPAR_WRITE_XML_MODE MSK_WRITE_XML_MODE_ROW file

Groups *Data input/output*

11.2.3 String parameters

MSK_SPAR_BAS_SOL_FILE_NAME

Name of the bas solution file.

Accepted Any valid file name.

Example mosek -d MSK_SPAR_BAS_SOL_FILE_NAME somevalue file

Groups *Data input/output, Solution input/output*

MSK_SPAR_DATA_FILE_NAME

Data are read and written to this file.

Accepted Any valid file name.

Example mosek -d MSK_SPAR_DATA_FILE_NAME somevalue file

Groups *Data input/output*

MSK_SPAR_DEBUG_FILE_NAME

MOSEK debug file.

Accepted Any valid file name.

Example mosek -d MSK_SPAR_DEBUG_FILE_NAME somevalue file

Groups *Data input/output*

MSK_SPAR_INT_SOL_FILE_NAME

Name of the int solution file.

Accepted Any valid file name.

Example mosek -d MSK_SPAR_INT_SOL_FILE_NAME somevalue file

Groups *Data input/output, Solution input/output*

MSK_SPAR_ITR_SOL_FILE_NAME

Name of the itr solution file.

Accepted Any valid file name.

Example mosek -d MSK_SPAR_ITR_SOL_FILE_NAME somevalue file

Groups *Data input/output, Solution input/output*

MSK_SPAR_MIO_DEBUG_STRING

For internal debugging purposes.

Accepted Any valid string.

Example mosek -d MSK_SPAR_MIO_DEBUG_STRING somevalue file

Groups *Data input/output*

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.

Default

%%

Accepted Any valid string.

Example mosek -d MSK_SPAR_PARAM_COMMENT_SIGN %% file

Groups *Data input/output*

MSK_SPAR_PARAM_READ_FILE_NAME

Modifications to the parameter database is read from this file.

Accepted Any valid file name.

Example mosek -d MSK_SPAR_PARAM_READ_FILE_NAME somevalue file

Groups *Data input/output*

MSK_SPAR_PARAM_WRITE_FILE_NAME

The parameter database is written to this file.

Accepted Any valid file name.

Example `mosek -d MSK_SPAR_PARAM_WRITE_FILE_NAME somevalue file`

Groups *Data input/output*

MSK_SPAR_READ_MPS_BOU_NAME

Name of the BOUNDS vector used. An empty name means that the first BOUNDS vector is used.

Accepted Any valid MPS name.

Example `mosek -d MSK_SPAR_READ_MPS_BOU_NAME somevalue file`

Groups *Data input/output*

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.

Accepted Any valid MPS name.

Example `mosek -d MSK_SPAR_READ_MPS_OBJ_NAME somevalue file`

Groups *Data input/output*

MSK_SPAR_READ_MPS_RAN_NAME

Name of the RANGE vector used. An empty name means that the first RANGE vector is used.

Accepted Any valid MPS name.

Example `mosek -d MSK_SPAR_READ_MPS_RAN_NAME somevalue file`

Groups *Data input/output*

MSK_SPAR_READ_MPS_RHS_NAME

Name of the RHS used. An empty name means that the first RHS vector is used.

Accepted Any valid MPS name.

Example `mosek -d MSK_SPAR_READ_MPS_RHS_NAME somevalue file`

Groups *Data input/output*

MSK_SPAR_REMOTE_OPTSERVER_HOST

URL of the remote optimization server in the format `(http|https)://server:port`. If set, all subsequent calls to any **MOSEK** function that involves synchronous optimization will be sent to the specified OptServer instead of being executed locally. Passing empty string deactivates this redirection.

Accepted Any valid URL.

Example `mosek -d MSK_SPAR_REMOTE_OPTSERVER_HOST somevalue file`

Groups *Overall system*

MSK_SPAR_REMOTE_TLS_CERT

List of known server certificates in PEM format.

Accepted PEM files separated by new-lines.

Example `mosek -d MSK_SPAR_REMOTE_TLS_CERT somevalue file`

Groups *Overall system*

MSK_SPAR_REMOTE_TLS_CERT_PATH

Path to known server certificates in PEM format.

Accepted Any valid path.

Example `mosek -d MSK_SPAR_REMOTE_TLS_CERT_PATH somevalue file`

Groups *Overall system*

MSK_SPAR_SENSITIVITY_FILE_NAME

Not applicable.

Accepted Any valid string.

Example `mosek -d MSK_SPAR_SENSITIVITY_FILE_NAME somevalue file`

Groups *Data input/output*

MSK_SPAR_SENSITIVITY_RES_FILE_NAME

Not applicable.

Accepted Any valid string.

Example `mosek -d MSK_SPAR_SENSITIVITY_RES_FILE_NAME somevalue file`

Groups *Data input/output*

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 $xc[i] \geq blc[i] + 0.5$ should be listed. An empty filter means that no filter is applied.

Accepted Any valid filter.

Example `mosek -d MSK_SPAR_SOL_FILTER_XC_LOW somevalue file`

Groups *Data input/output, Solution input/output*

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.

Accepted Any valid filter.

Example `mosek -d MSK_SPAR_SOL_FILTER_XC_UPR somevalue file`

Groups *Data input/output, Solution input/output*

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.

Accepted Any valid filter.

Example `mosek -d MSK_SPAR_SOL_FILTER_XX_LOW somevalue file`

Groups *Data input/output, Solution input/output*

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.

Accepted Any valid file name.

Example `mosek -d MSK_SPAR_SOL_FILTER_XX_UPR somevalue file`

Groups *Data input/output, Solution input/output*

MSK_SPAR_STAT_KEY

Key used when writing the summary file.

Accepted Any valid string.

Example `mosek -d MSK_SPAR_STAT_KEY somevalue file`

Groups *Data input/output*

MSK_SPAR_STAT_NAME

Name used when writing the statistics file.

Accepted Any valid XML string.

Example `mosek -d MSK_SPAR_STAT_NAME somevalue file`

Groups *Data input/output*

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.

Default `xmskgen`

Accepted Any valid string.

Example `mosek -d MSK_SPAR_WRITE_LP_GEN_VAR_NAME xmskgen file`

Groups *Data input/output*

11.3 Response codes

Response codes include:

- *Termination codes*
- *Warnings*
- *Errors*

The numerical code (in brackets) identifies the response in error messages and in the log output.

11.3.1 Termination

MSK_RES_OK (0)

No error occurred.

MSK_RES_TRM_MAX_ITERATIONS (100000)

The optimizer terminated at the maximum number of iterations.

MSK_RES_TRM_MAX_TIME (100001)

The optimizer terminated at the maximum amount of time.

MSK_RES_TRM_OBJECTIVE_RANGE (100002)

The optimizer terminated with an objective value outside the objective range.

MSK_RES_TRM_MIO_NUM_RELAXS (100008)

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

MSK_RES_TRM_MIO_NUM_BRANCHES (100009)

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

MSK_RES_TRM_NUM_MAX_NUM_INT_SOLUTIONS (100015)

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

MSK_RES_TRM_STALL (100006)

The optimizer is terminated due to slow progress.

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

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

Some common causes of stalling are a) badly scaled models, b) near feasible or near infeasible problems.

MSK_RES_TRM_USER_CALLBACK (100007)

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

MSK_RES_TRM_MAX_NUM_SETBACKS (100020)

The optimizer terminated as the maximum number of set-backs was reached. This indicates serious numerical problems and a possibly badly formulated problem.

MSK_RES_TRM_NUMERICAL_PROBLEM (100025)

The optimizer terminated due to numerical problems.

MSK_RES_TRM_LOST_RACE (100027)

Lost a race.

MSK_RES_TRM_INTERNAL (100030)

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

MSK_RES_TRM_INTERNAL_STOP (100031)

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

11.3.2 Warnings

MSK_RES_WRN_OPEN_PARAM_FILE (50)

The parameter file could not be opened.

MSK_RES_WRN_LARGE_BOUND (51)

A numerically large bound value is specified.

MSK_RES_WRN_LARGE_LO_BOUND (52)

A numerically large lower bound value is specified.

MSK_RES_WRN_LARGE_UP_BOUND (53)

A numerically large upper bound value is specified.

MSK_RES_WRN_LARGE_CON_FX (54)

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

MSK_RES_WRN_LARGE_CJ (57)

A numerically large value is specified for one c_j .

MSK_RES_WRN_LARGE_AIJ (62)

A numerically large value is specified for an $a_{i,j}$ element in A . The parameter [*MSK_DPAR_DATA_TOL_AIJ_LARGE*](#) controls when an $a_{i,j}$ is considered large.

MSK_RES_WRN_ZERO_AIJ (63)

One or more zero elements are specified in A .

MSK_RES_WRN_NAME_MAX_LEN (65)

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

MSK_RES_WRN_SPAR_MAX_LEN (66)

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

MSK_RES_WRN_MPS_SPLIT_RHS_VECTOR (70)

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

MSK_RES_WRN_MPS_SPLIT_RAN_VECTOR (71)

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

MSK_RES_WRN_MPS_SPLIT_BOU_VECTOR (72)

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

MSK_RES_WRN_LP_OLD_QUAD_FORMAT (80)

Missing $\sqrt{}$ after quadratic expressions in bound or objective.

MSK_RES_WRN_LP_DROP_VARIABLE (85)

Ignored a variable because the variable was not previously defined. Usually this implies that a variable appears in the bound section but not in the objective or the constraints.

MSK_RES_WRN_NZ_IN_UPR_TRI (200)

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

MSK_RES_WRN_DROPPED_NZ_QOBJ (201)

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

MSK_RES_WRN_IGNORE_INTEGER (250)

Ignored integer constraints.

MSK_RES_WRN_NO_GLOBAL_OPTIMIZER (251)

No global optimizer is available.

MSK_RES_WRN_MIO_INFEASIBLE_FINAL (270)

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

MSK_RES_WRN_SOL_FILTER (300)

Invalid solution filter is specified.

MSK_RES_WRN_UNDEF_SOL_FILE_NAME (350)

Undefined name occurred in a solution.

MSK_RES_WRN_SOL_FILE_IGNORED_CON (351)
 One or more lines in the constraint section were ignored when reading a solution file.

MSK_RES_WRN_SOL_FILE_IGNORED_VAR (352)
 One or more lines in the variable section were ignored when reading a solution file.

MSK_RES_WRN_TOO_FEW_BASIS_VARS (400)
 An incomplete basis has been specified. Too few basis variables are specified.

MSK_RES_WRN_TOO_MANY_BASIS_VARS (405)
 A basis with too many variables has been specified.

MSK_RES_WRN_LICENSE_EXPIRE (500)
 The license expires.

MSK_RES_WRN_LICENSE_SERVER (501)
 The license server is not responding.

MSK_RES_WRN_EMPTY_NAME (502)
 A variable or constraint name is empty. The output file may be invalid.

MSK_RES_WRN_USING_GENERIC_NAMES (503)
 Generic names are used because a name is not valid. For instance when writing an LP file the names must not contain blanks or start with a digit.

MSK_RES_WRN_INVALID_MPS_NAME (504)
 A name e.g. a row name is not a valid MPS name.

MSK_RES_WRN_INVALID_MPS_OBJ_NAME (505)
 The objective name is not a valid MPS name.

MSK_RES_WRN_LICENSE_FEATURE_EXPIRE (509)
 The license expires.

MSK_RES_WRN_PARAM_NAME_DOUB (510)
 The parameter name is not recognized as a double parameter.

MSK_RES_WRN_PARAM_NAME_INT (511)
 The parameter name is not recognized as an integer parameter.

MSK_RES_WRN_PARAM_NAME_STR (512)
 The parameter name is not recognized as a string parameter.

MSK_RES_WRN_PARAM_STR_VALUE (515)
 The string is not recognized as a symbolic value for the parameter.

MSK_RES_WRN_PARAM_IGNORED_CMIO (516)
 A parameter was ignored by the conic mixed integer optimizer.

MSK_RES_WRN_ZEROS_IN_SPARSE_ROW (705)
 One or more (near) zero elements are specified in a sparse row of a matrix. Since, it is redundant to specify zero elements then it may indicate an error.

MSK_RES_WRN_ZEROS_IN_SPARSE_COL (710)
 One or more (near) zero elements are specified in a sparse column of a matrix. It is redundant to specify zero elements. Hence, it may indicate an error.

MSK_RES_WRN_INCOMPLETE_LINEAR_DEPENDENCY_CHECK (800)
 The linear dependency check(s) is incomplete. Normally this is not an important warning unless the optimization problem has been formulated with linear dependencies. Linear dependencies may prevent **MOSEK** from solving the problem.

MSK_RES_WRN_ELIMINATOR_SPACE (801)
 The eliminator is skipped at least once due to lack of space.

MSK_RES_WRN_PRESOLVE_OUTOFSPACE (802)
 The presolve is incomplete due to lack of space.

MSK_RES_WRN_PRESOLVE_PRIMAL_PERTUBATIONS (803)
 The presolve perturbed the bounds of the primal problem. This is an indication that the problem is nearly infeasible.

MSK_RES_WRN_WRITE_CHANGED_NAMES (830)
 Some names were changed because they were invalid for the output file format.

MSK_RES_WRN_WRITE_DISCARDED_CFIX (831)
 The fixed objective term could not be converted to a variable and was discarded in the output file.

MSK_RES_WRN_DUPLICATE_CONSTRAINT_NAMES (850)
 Two constraint names are identical.

MSK_RES_WRN_DUPLICATE_VARIABLE_NAMES (851)
 Two variable names are identical.

MSK_RES_WRN_DUPLICATE_BARVARIABLE_NAMES (852)
Two barvariable names are identical.

MSK_RES_WRN_DUPLICATE_CONE_NAMES (853)
Two cone names are identical.

MSK_RES_WRN_WRITE_LP_INVALID_VAR_NAMES (854)
LP file will be written with generic variable names.

MSK_RES_WRN_WRITE_LP_DUPLICATE_VAR_NAMES (855)
LP file will be written with generic variable names.

MSK_RES_WRN_WRITE_LP_INVALID_CON_NAMES (856)
LP file will be written with generic constraint names.

MSK_RES_WRN_WRITE_LP_DUPLICATE_CON_NAMES (857)
LP file will be written with generic constraint names.

MSK_RES_WRN_ANA_LARGE_BOUNDS (900)
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 $+\text{inf}$ or $-\text{inf}$.

MSK_RES_WRN_ANA_C_ZERO (901)
This warning is issued by the problem analyzer, if the coefficients in the linear part of the objective are all zero.

MSK_RES_WRN_ANA_EMPTY_COLS (902)
This warning is issued by the problem analyzer, if columns, in which all coefficients are zero, are found.

MSK_RES_WRN_ANA_CLOSE_BOUNDS (903)
This warning is issued by problem analyzer, if ranged constraints or variables with very close upper and lower bounds are detected. One should consider treating such constraints as equalities and such variables as constants.

MSK_RES_WRN_ANA_ALMOST_INT_BOUNDS (904)
This warning is issued by the problem analyzer if a constraint is bound nearly integral.

MSK_RES_WRN_NO_INFEASIBILITY_REPORT_WHEN_MATRIX_VARIABLES (930)
An infeasibility report is not available when the problem contains matrix variables.

MSK_RES_WRN_NO_DUALIZER (950)
No automatic dualizer is available for the specified problem. The primal problem is solved.

MSK_RES_WRN_SYM_MAT_LARGE (960)
A numerically large value is specified for an $e_{i,j}$ element in E . The parameter `MSK_DPAR_DATA_SYM_MAT_TOL_LARGE` controls when an $e_{i,j}$ is considered large.

MSK_RES_WRN_MODIFIED_DOUBLE_PARAMETER (970)
A double parameter related to solver tolerances has a non-default value.

MSK_RES_WRN_LARGE_FIJ (980)
A numerically large value is specified for an $f_{i,j}$ element in F . The parameter `MSK_DPAR_DATA_TOL_AIJ_LARGE` controls when an $f_{i,j}$ is considered large.

11.3.3 Errors

MSK_RES_ERR_LICENSE (1000)
Invalid license.

MSK_RES_ERR_LICENSE_EXPIRED (1001)
The license has expired.

MSK_RES_ERR_LICENSE_VERSION (1002)
The license is valid for another version of **MOSEK**.

MSK_RES_ERR_LICENSE_OLD_SERVER_VERSION (1003)
The version of the FlexLM license server is too old. You should upgrade the license server to one matching this version of **MOSEK**. It will support this and all older versions of **MOSEK**.

This error can appear if the client was updated to a new version which includes an upgrade of the licensing module, making it incompatible with a much older license server.

MSK_RES_ERR_SIZE_LICENSE (1005)
The problem is bigger than the license.

MSK_RES_ERR_PROB_LICENSE (1006)
The software is not licensed to solve the problem.

MSK_RES_ERR_FILE_LICENSE (1007)
Invalid license file.

MSK_RES_ERR_MISSING_LICENSE_FILE (1008)
MOSEK cannot find license file or a token server. See the **MOSEK** licensing manual for details.

MSK_RES_ERR_SIZE_LICENSE_CON (1010)
The problem has too many constraints to be solved with the available license.

MSK_RES_ERR_SIZE_LICENSE_VAR (1011)
The problem has too many variables to be solved with the available license.

MSK_RES_ERR_SIZE_LICENSE_INTVAR (1012)
The problem contains too many integer variables to be solved with the available license.

MSK_RES_ERR_OPTIMIZER_LICENSE (1013)
The optimizer required is not licensed.

MSK_RES_ERR_FLEXLM (1014)
The FLEXlm license manager reported an error.

MSK_RES_ERR_LICENSE_SERVER (1015)
The license server is not responding.

MSK_RES_ERR_LICENSE_MAX (1016)
Maximum number of licenses is reached.

MSK_RES_ERR_LICENSE_MOSEKLM_DAEMON (1017)
The MOSEKLM license manager daemon is not up and running.

MSK_RES_ERR_LICENSE_FEATURE (1018)
A requested feature is not available in the license file(s). Most likely due to an incorrect license system setup.

MSK_RES_ERR_PLATFORM_NOT_LICENSED (1019)
A requested license feature is not available for the required platform.

MSK_RES_ERR_LICENSE_CANNOT_ALLOCATE (1020)
The license system cannot allocate the memory required.

MSK_RES_ERR_LICENSE_CANNOT_CONNECT (1021)
MOSEK cannot connect to the license server. Most likely the license server is not up and running.

MSK_RES_ERR_LICENSE_INVALID_HOSTID (1025)
The host ID specified in the license file does not match the host ID of the computer.

MSK_RES_ERR_LICENSE_SERVER_VERSION (1026)
The version specified in the checkout request is greater than the highest version number the daemon supports.

MSK_RES_ERR_LICENSE_NO_SERVER_SUPPORT (1027)
The license server does not support the requested feature. Possible reasons for this error include:

- The feature has expired.
- The feature's start date is later than today's date.
- The version requested is higher than feature's the highest supported version.
- A corrupted license file.

Try restarting the license and inspect the license server debug file, usually called `lmgrd.log`.

MSK_RES_ERR_LICENSE_NO_SERVER_LINE (1028)
There is no **SERVER** line in the license file. All non-zero license count features need at least one **SERVER** line.

MSK_RES_ERR_OLDER_DLL (1035)
The dynamic link library is older than the specified version.

MSK_RES_ERR_NEWER_DLL (1036)
The dynamic link library is newer than the specified version.

MSK_RES_ERR_LINK_FILE_DLL (1040)
A file cannot be linked to a stream in the DLL version.

MSK_RES_ERR_THREAD_MUTEX_INIT (1045)
Could not initialize a mutex.

MSK_RES_ERR_THREAD_MUTEX_LOCK (1046)
Could not lock a mutex.

MSK_RES_ERR_THREAD_MUTEX_UNLOCK (1047)
Could not unlock a mutex.

MSK_RES_ERR_THREAD_CREATE (1048)
 Could not create a thread. This error may occur if a large number of environments are created and not deleted again. In any case it is a good practice to minimize the number of environments created.

MSK_RES_ERR_THREAD_COND_INIT (1049)
 Could not initialize a condition.

MSK_RES_ERR_UNKNOWN (1050)
 Unknown error.

MSK_RES_ERR_SPACE (1051)
 Out of space.

MSK_RES_ERR_FILE_OPEN (1052)
 Error while opening a file.

MSK_RES_ERR_FILE_READ (1053)
 File read error.

MSK_RES_ERR_FILE_WRITE (1054)
 File write error.

MSK_RES_ERR_DATA_FILE_EXT (1055)
 The data file format cannot be determined from the file name.

MSK_RES_ERR_INVALID_FILE_NAME (1056)
 An invalid file name has been specified.

MSK_RES_ERR_INVALID_SOL_FILE_NAME (1057)
 An invalid file name has been specified.

MSK_RES_ERR_END_OF_FILE (1059)
 End of file reached.

MSK_RES_ERR_NULL_ENV (1060)
`env` is a NULL pointer.

MSK_RES_ERR_NULL_TASK (1061)
`task` is a NULL pointer.

MSK_RES_ERR_INVALID_STREAM (1062)
 An invalid stream is referenced.

MSK_RES_ERR_NO_INIT_ENV (1063)
`env` is not initialized.

MSK_RES_ERR_INVALID_TASK (1064)
 The `task` is invalid.

MSK_RES_ERR_NULL_POINTER (1065)
 An argument to a function is unexpectedly a NULL pointer.

MSK_RES_ERR_LIVING_TASKS (1066)
 All tasks associated with an environment must be deleted before the environment is deleted. There are still some undeleted tasks.

MSK_RES_ERR_BLANK_NAME (1070)
 An all blank name has been specified.

MSK_RES_ERR_DUP_NAME (1071)
 The same name was used multiple times for the same problem item type.

MSK_RES_ERR_FORMAT_STRING (1072)
 The name format string is invalid.

MSK_RES_ERR_SPARSITY_SPECIFICATION (1073)
 The sparsity included an index that was out of bounds of the shape.

MSK_RES_ERR_MISMATCHING_DIMENSION (1074)
 Mismatching dimensions specified in arguments

MSK_RES_ERR_INVALID_OBJ_NAME (1075)
 An invalid objective name is specified.

MSK_RES_ERR_INVALID_CON_NAME (1076)
 An invalid constraint name is used.

MSK_RES_ERR_INVALID_VAR_NAME (1077)
 An invalid variable name is used.

MSK_RES_ERR_INVALID_CONE_NAME (1078)
 An invalid cone name is used.

MSK_RES_ERR_INVALID_BARVAR_NAME (1079)
 An invalid symmetric matrix variable name is used.

MSK_RES_ERR_SPACE_LEAKING (1080)
MOSEK is leaking memory. This can be due to either an incorrect use of **MOSEK** or a bug.

MSK_RES_ERR_SPACE_NO_INFO (1081)
 No available information about the space usage.

MSK_RES_ERR_DIMENSION_SPECIFICATION (1082)
 Invalid dimension specification

MSK_RES_ERR_AXIS_NAME_SPECIFICATION (1083)
 Invalid axis names specification

MSK_RES_ERR_READ_FORMAT (1090)
 The specified format cannot be read.

MSK_RES_ERR_MPS_FILE (1100)
 An error occurred while reading an MPS file.

MSK_RES_ERR_MPS_INV_FIELD (1101)
 A field in the MPS file is invalid. Probably it is too wide.

MSK_RES_ERR_MPS_INV_MARKER (1102)
 An invalid marker has been specified in the MPS file.

MSK_RES_ERR_MPS_NULL_CON_NAME (1103)
 An empty constraint name is used in an MPS file.

MSK_RES_ERR_MPS_NULL_VAR_NAME (1104)
 An empty variable name is used in an MPS file.

MSK_RES_ERR_MPS_UNDEF_CON_NAME (1105)
 An undefined constraint name occurred in an MPS file.

MSK_RES_ERR_MPS_UNDEF_VAR_NAME (1106)
 An undefined variable name occurred in an MPS file.

MSK_RES_ERR_MPS_INVALID_CON_KEY (1107)
 An invalid constraint key occurred in an MPS file.

MSK_RES_ERR_MPS_INVALID_BOUND_KEY (1108)
 An invalid bound key occurred in an MPS file.

MSK_RES_ERR_MPS_INVALID_SEC_NAME (1109)
 An invalid section name occurred in an MPS file.

MSK_RES_ERR_MPS_NO_OBJECTIVE (1110)
 No objective is defined in an MPS file.

MSK_RES_ERR_MPS_SPLITTED_VAR (1111)
 All elements in a column of the A matrix must be specified consecutively. Hence, it is illegal to specify non-zero elements in A for variable 1, then for variable 2 and then variable 1 again.

MSK_RES_ERR_MPS_MUL_CON_NAME (1112)
 A constraint name was specified multiple times in the ROWS section.

MSK_RES_ERR_MPS_MUL_QSEC (1113)
 Multiple QSECTIONS are specified for a constraint in the MPS data file.

MSK_RES_ERR_MPS_MUL_QOBJ (1114)
 The Q term in the objective is specified multiple times in the MPS data file.

MSK_RES_ERR_MPS_INV_SEC_ORDER (1115)
 The sections in the MPS data file are not in the correct order.

MSK_RES_ERR_MPS_MUL_CSEC (1116)
 Multiple CSECTIONS are given the same name.

MSK_RES_ERR_MPS_CONE_TYPE (1117)
 Invalid cone type specified in a CSECTION.

MSK_RES_ERR_MPS_CONE_OVERLAP (1118)
 A variable is specified to be a member of several cones.

MSK_RES_ERR_MPS_CONE_REPEAT (1119)
 A variable is repeated within the CSECTION.

MSK_RES_ERR_MPS_NON_SYMMETRIC_Q (1120)
 A non symmetric matrix has been specified.

MSK_RES_ERR_MPS_DUPLICATE_Q_ELEMENT (1121)
 Duplicate elements is specified in a Q matrix.

MSK_RES_ERR_MPS_INVALID_OBJSENSE (1122)
 An invalid objective sense is specified.

MSK_RES_ERR_MPS_TAB_IN_FIELD2 (1125)
 A tab char occurred in field 2.

MSK_RES_ERR_MPS_TAB_IN_FIELD3 (1126)
 A tab char occurred in field 3.

MSK_RES_ERR_MPS_TAB_IN_FIELD5 (1127)
 A tab char occurred in field 5.

MSK_RES_ERR_MPS_INVALID_OBJ_NAME (1128)
 An invalid objective name is specified.

MSK_RES_ERR_MPS_INVALID_KEY (1129)
 An invalid indicator key occurred in an MPS file.

MSK_RES_ERR_MPS_INVALID_INDICATOR_CONSTRAINT (1130)
 An invalid indicator constraint is used. It must not be a ranged constraint.

MSK_RES_ERR_MPS_INVALID_INDICATOR_VARIABLE (1131)
 An invalid indicator variable is specified. It must be a binary variable.

MSK_RES_ERR_MPS_INVALID_INDICATOR_VALUE (1132)
 An invalid indicator value is specified. It must be either 0 or 1.

MSK_RES_ERR_MPS_INVALID_INDICATOR_QUADRATIC_CONSTRAINT (1133)
 A quadratic constraint can be be an indicator constraint.

MSK_RES_ERR_OPF_SYNTAX (1134)
 Syntax error in an OPF file

MSK_RES_ERR_OPF_PREMATURE_EOF (1136)
 Premature end of file in an OPF file.

MSK_RES_ERR_OPF_MISMATCHED_TAG (1137)
 Mismatched end-tag in OPF file

MSK_RES_ERR_OPF_DUPLICATE_BOUND (1138)
 Either upper or lower bound was specified twice in OPF file

MSK_RES_ERR_OPF_DUPLICATE_CONSTRAINT_NAME (1139)
 Duplicate constraint name in OPF File

MSK_RES_ERR_OPF_INVALID_CONE_TYPE (1140)
 Invalid cone type in OPF File

MSK_RES_ERR_OPF_INCORRECT_TAG_PARAM (1141)
 Invalid number of parameters in start-tag in OPF File

MSK_RES_ERR_OPF_INVALID_TAG (1142)
 Invalid start-tag in OPF File

MSK_RES_ERR_OPF_DUPLICATE_CONE_ENTRY (1143)
 Same variable appears in multiple cones in OPF File

MSK_RES_ERR_OPF_TOO_LARGE (1144)
 The problem is too large to be correctly loaded

MSK_RES_ERR_OPF_DUAL_INTEGER_SOLUTION (1146)
 Dual solution values are not allowed in OPF File

MSK_RES_ERR_LP_INCOMPATIBLE (1150)
 The problem cannot be written to an LP formatted file.

MSK_RES_ERR_LP_EMPTY (1151)
 The problem cannot be written to an LP formatted file.

MSK_RES_ERR_LP_DUP_SLACK_NAME (1152)
 The name of the slack variable added to a ranged constraint already exists.

MSK_RES_ERR_WRITE_MPS_INVALID_NAME (1153)
 An invalid name is created while writing an MPS file. Usually this will make the MPS file unreadable.

MSK_RES_ERR_LP_INVALID_VAR_NAME (1154)
 A variable name is invalid when used in an LP formatted file.

MSK_RES_ERR_LP_FREE_CONSTRAINT (1155)
 Free constraints cannot be written in LP file format.

MSK_RES_ERR_WRITE_OPF_INVALID_VAR_NAME (1156)
 Empty variable names cannot be written to OPF files.

MSK_RES_ERR_LP_FILE_FORMAT (1157)
 Syntax error in an LP file.

MSK_RES_ERR_WRITE_LP_FORMAT (1158)
 Problem cannot be written as an LP file.

MSK_RES_ERR_READ_LP_MISSING_END_TAG (1159)
 Syntax error in LP file. Possibly missing End tag.

MSK_RES_ERR_LP_INDICATOR_VAR (1160)
 An indicator variable was not declared binary

MSK_RES_ERR_WRITE_LP_NON_UNIQUE_NAME (1161)
 An auto-generated name is not unique.

MSK_RES_ERR_READ_LP_NONEXISTING_NAME (1162)
 A variable never occurred in objective or constraints.

MSK_RES_ERR_LP_WRITE_CONIC_PROBLEM (1163)
 The problem contains cones that cannot be written to an LP formatted file.

MSK_RES_ERR_LP_WRITE_GECO_PROBLEM (1164)
 The problem contains general convex terms that cannot be written to an LP formatted file.

MSK_RES_ERR_WRITING_FILE (1166)
 An error occurred while writing file

MSK_RES_ERR_INVALID_NAME_IN_SOL_FILE (1170)
 An invalid name occurred in a solution file.

MSK_RES_ERR_LP_INVALID_CON_NAME (1171)
 A constraint name is invalid when used in an LP formatted file.

MSK_RES_ERR_JSON_SYNTAX (1175)
 Syntax error in an JSON data

MSK_RES_ERR_JSON_STRING (1176)
 Error in JSON string.

MSK_RES_ERR_JSON_NUMBER_OVERFLOW (1177)
 Invalid number entry - wrong type or value overflow.

MSK_RES_ERR_JSON_FORMAT (1178)
 Error in an JSON Task file

MSK_RES_ERR_JSON_DATA (1179)
 Inconsistent data in JSON Task file

MSK_RES_ERR_JSON_MISSING_DATA (1180)
 Missing data section in JSON task file.

MSK_RES_ERR_PTF_INCOMPATIBILITY (1181)
 Incompatible item

MSK_RES_ERR_PTF_UNDEFINED_ITEM (1182)
 Undefined symbol referenced

MSK_RES_ERR_PTF_INCONSISTENCY (1183)
 Inconsistent size of item

MSK_RES_ERR_PTF_FORMAT (1184)
 Syntax error in an PTF file

MSK_RES_ERR_ARGUMENT_LENNEQ (1197)
 Incorrect length of arguments.

MSK_RES_ERR_ARGUMENT_TYPE (1198)
 Incorrect argument type.

MSK_RES_ERR_NUM_ARGUMENTS (1199)
 Incorrect number of function arguments.

MSK_RES_ERR_IN_ARGUMENT (1200)
 A function argument is incorrect.

MSK_RES_ERR_ARGUMENT_DIMENSION (1201)
 A function argument is of incorrect dimension.

MSK_RES_ERR_SHAPE_IS_TOO_LARGE (1202)
 The size of the n-dimensional shape is too large.

MSK_RES_ERR_INDEX_IS_TOO_SMALL (1203)
 An index in an argument is too small.

MSK_RES_ERR_INDEX_IS_TOO_LARGE (1204)
 An index in an argument is too large.

MSK_RES_ERR_INDEX_IS_NOT_UNIQUE (1205)
 An index in an argument is is unique.

MSK_RES_ERR_PARAM_NAME (1206)
 The parameter name is not correct.

MSK_RES_ERR_PARAM_NAME_DOU (1207)
 The parameter name is not correct for a double parameter.

MSK_RES_ERR_PARAM_NAME_INT (1208)
 The parameter name is not correct for an integer parameter.

MSK_RES_ERR_PARAM_NAME_STR (1209)
 The parameter name is not correct for a string parameter.

MSK_RES_ERR_PARAM_INDEX (1210)
 Parameter index is out of range.

MSK_RES_ERR_PARAM_IS_TOO_LARGE (1215)
 The parameter value is too large.

MSK_RES_ERR_PARAM_IS_TOO_SMALL (1216)
 The parameter value is too small.

MSK_RES_ERR_PARAM_VALUE_STR (1217)
 The parameter value string is incorrect.

MSK_RES_ERR_PARAM_TYPE (1218)
 The parameter type is invalid.

MSK_RES_ERR_INF_DOU_INDEX (1219)
 A double information index is out of range for the specified type.

MSK_RES_ERR_INF_INT_INDEX (1220)
 An integer information index is out of range for the specified type.

MSK_RES_ERR_INDEX_ARR_IS_TOO_SMALL (1221)
 An index in an array argument is too small.

MSK_RES_ERR_INDEX_ARR_IS_TOO_LARGE (1222)
 An index in an array argument is too large.

MSK_RES_ERR_INF_LINT_INDEX (1225)
 A long integer information index is out of range for the specified type.

MSK_RES_ERR_ARG_IS_TOO_SMALL (1226)
 The value of a argument is too small.

MSK_RES_ERR_ARG_IS_TOO_LARGE (1227)
 The value of a argument is too large.

MSK_RES_ERR_INVALID_WHICHSOL (1228)
 whichsol is invalid.

MSK_RES_ERR_INF_DOU_NAME (1230)
 A double information name is invalid.

MSK_RES_ERR_INF_INT_NAME (1231)
 An integer information name is invalid.

MSK_RES_ERR_INF_TYPE (1232)
 The information type is invalid.

MSK_RES_ERR_INF_LINT_NAME (1234)
 A long integer information name is invalid.

MSK_RES_ERR_INDEX (1235)
 An index is out of range.

MSK_RES_ERR_WHICHSOL (1236)
 The solution defined by whichsol does not exists.

MSK_RES_ERR_SOLITEM (1237)
 The solution item number solitem is invalid. Please note that *MSK_SOL_ITEM_SNX* is invalid for the basic solution.

MSK_RES_ERR_WHICHITEM_NOT_ALLOWED (1238)
 whichitem is unacceptable.

MSK_RES_ERR_MAXNUMCON (1240)
 The maximum number of constraints specified is smaller than the number of constraints in the task.

MSK_RES_ERR_MAXNUMVAR (1241)
 The maximum number of variables specified is smaller than the number of variables in the task.

MSK_RES_ERR_MAXNUMBARVAR (1242)
The maximum number of semidefinite variables specified is smaller than the number of semidefinite variables in the task.

MSK_RES_ERR_MAXNUMQNZ (1243)
The maximum number of non-zeros specified for the Q matrices is smaller than the number of non-zeros in the current Q matrices.

MSK_RES_ERR_TOO_SMALL_MAX_NUM_NZ (1245)
The maximum number of non-zeros specified is too small.

MSK_RES_ERR_INVALID_IDX (1246)
A specified index is invalid.

MSK_RES_ERR_INVALID_MAX_NUM (1247)
A specified index is invalid.

MSK_RES_ERR_UNALLOWED_WHICH_SOL (1248)
The value of `whichsol` is not allowed.

MSK_RES_ERR_NUMCONLIM (1250)
Maximum number of constraints limit is exceeded.

MSK_RES_ERR_NUMVARLIM (1251)
Maximum number of variables limit is exceeded.

MSK_RES_ERR_TOO_SMALL_MAXNUMANZ (1252)
The maximum number of non-zeros specified for A is smaller than the number of non-zeros in the current A .

MSK_RES_ERR_INV_APTRE (1253)
`aptre[j]` is strictly smaller than `aptrb[j]` for some j .

MSK_RES_ERR_MUL_A_ELEMENT (1254)
An element in A is defined multiple times.

MSK_RES_ERR_INV_BK (1255)
Invalid bound key.

MSK_RES_ERR_INV_BKC (1256)
Invalid bound key is specified for a constraint.

MSK_RES_ERR_INV_BKX (1257)
An invalid bound key is specified for a variable.

MSK_RES_ERR_INV_VAR_TYPE (1258)
An invalid variable type is specified for a variable.

MSK_RES_ERR_SOLVER_PROBTYPE (1259)
Problem type does not match the chosen optimizer.

MSK_RES_ERR_OBJECTIVE_RANGE (1260)
Empty objective range.

MSK_RES_ERR_BASIS (1266)
An invalid basis is specified. Either too many or too few basis variables are specified.

MSK_RES_ERR_INV_SKC (1267)
Invalid value in `skc`.

MSK_RES_ERR_INV_SKX (1268)
Invalid value in `skx`.

MSK_RES_ERR_INV_SKN (1274)
Invalid value in `skn`.

MSK_RES_ERR_INV_SK_STR (1269)
Invalid status key string encountered.

MSK_RES_ERR_INV_SK (1270)
Invalid status key code.

MSK_RES_ERR_INV_CONE_TYPE_STR (1271)
Invalid cone type string encountered.

MSK_RES_ERR_INV_CONE_TYPE (1272)
Invalid cone type code is encountered.

MSK_RES_ERR_INVALID_SURPLUS (1275)
Invalid surplus.

MSK_RES_ERR_INV_NAME_ITEM (1280)
An invalid name item code is used.

MSK_RES_ERR_PRO_ITEM (1281)
An invalid problem is used.

MSK_RES_ERR_INVALID_FORMAT_TYPE (1283)
Invalid format type.

MSK_RES_ERR_FIRSTI (1285)
Invalid firsti.

MSK_RES_ERR_LASTI (1286)
Invalid lasti.

MSK_RES_ERR_FIRSTJ (1287)
Invalid firstj.

MSK_RES_ERR_LASTJ (1288)
Invalid lastj.

MSK_RES_ERR_MAX_LEN_IS_TOO_SMALL (1289)
A maximum length that is too small has been specified.

MSK_RES_ERR_NONLINEAR_EQUALITY (1290)
The model contains a nonlinear equality which defines a nonconvex set.

MSK_RES_ERR_NONCONVEX (1291)
The optimization problem is nonconvex.

MSK_RES_ERR_NONLINEAR_RANGED (1292)
Nonlinear constraints with finite lower and upper bound always define a nonconvex feasible set.

MSK_RES_ERR_CON_Q_NOT_PSD (1293)
The quadratic constraint matrix is not positive semidefinite as expected for a constraint with finite upper bound. This results in a nonconvex problem. The parameter `MSK_DPAR_CHECK_CONVEXITY_REL_TOL` can be used to relax the convexity check.

MSK_RES_ERR_CON_Q_NOT_NSD (1294)
The quadratic constraint matrix is not negative semidefinite as expected for a constraint with finite lower bound. This results in a nonconvex problem. The parameter `MSK_DPAR_CHECK_CONVEXITY_REL_TOL` can be used to relax the convexity check.

MSK_RES_ERR_OBJ_Q_NOT_PSD (1295)
The quadratic coefficient matrix in the objective is not positive semidefinite as expected for a minimization problem. The parameter `MSK_DPAR_CHECK_CONVEXITY_REL_TOL` can be used to relax the convexity check.

MSK_RES_ERR_OBJ_Q_NOT_NSD (1296)
The quadratic coefficient matrix in the objective is not negative semidefinite as expected for a maximization problem. The parameter `MSK_DPAR_CHECK_CONVEXITY_REL_TOL` can be used to relax the convexity check.

MSK_RES_ERR_ARGUMENT_PERM_ARRAY (1299)
An invalid permutation array is specified.

MSK_RES_ERR_CONE_INDEX (1300)
An index of a non-existing cone has been specified.

MSK_RES_ERR_CONE_SIZE (1301)
A cone with incorrect number of members is specified.

MSK_RES_ERR_CONE_OVERLAP (1302)
One or more of the variables in the cone to be added is already member of another cone. Now assume the variable is x_j then add a new variable say x_k and the constraint

$$x_j = x_k$$

and then let x_k be member of the cone to be appended.

MSK_RES_ERR_CONE_REP_VAR (1303)
A variable is included multiple times in the cone.

MSK_RES_ERR_MAXNUMCONE (1304)
The value specified for `maxnumcone` is too small.

MSK_RES_ERR_CONE_TYPE (1305)
Invalid cone type specified.

MSK_RES_ERR_CONE_TYPE_STR (1306)
Invalid cone type specified.

MSK_RES_ERR_CONE_OVERLAP_APPEND (1307)
The cone to be appended has one variable which is already member of another cone.

MSK_RES_ERR_REMOVE_CONE_VARIABLE (1310)
 A variable cannot be removed because it will make a cone invalid.

MSK_RES_ERR_APPENDING_TOO_BIG_CONE (1311)
 Trying to append a too big cone.

MSK_RES_ERR_CONE_PARAMETER (1320)
 An invalid cone parameter.

MSK_RES_ERR_SOL_FILE_INVALID_NUMBER (1350)
 An invalid number is specified in a solution file.

MSK_RES_ERR_HUGE_C (1375)
 A huge value in absolute size is specified for one c_j .

MSK_RES_ERR_HUGE_AIJ (1380)
 A numerically huge value is specified for an $a_{i,j}$ element in A . The parameter `MSK_DPAR_DATA_TOL_AIJ_HUGE` controls when an $a_{i,j}$ is considered huge.

MSK_RES_ERR_DUPLICATE_AIJ (1385)
 An element in the A matrix is specified twice.

MSK_RES_ERR_LOWER_BOUND_IS_A_NAN (1390)
 The lower bound specified is not a number (nan).

MSK_RES_ERR_UPPER_BOUND_IS_A_NAN (1391)
 The upper bound specified is not a number (nan).

MSK_RES_ERR_INFINITE_BOUND (1400)
 A numerically huge bound value is specified.

MSK_RES_ERR_INV_QOBJ_SUBI (1401)
 Invalid value in `qosubi`.

MSK_RES_ERR_INV_QOBJ_SUBJ (1402)
 Invalid value in `qosubj`.

MSK_RES_ERR_INV_QOBJ_VAL (1403)
 Invalid value in `qoval`.

MSK_RES_ERR_INV_QCON_SUBK (1404)
 Invalid value in `qcsbk`.

MSK_RES_ERR_INV_QCON_SUBI (1405)
 Invalid value in `qcsubi`.

MSK_RES_ERR_INV_QCON_SUBJ (1406)
 Invalid value in `qcsbj`.

MSK_RES_ERR_INV_QCON_VAL (1407)
 Invalid value in `qcval`.

MSK_RES_ERR_QCON_SUBI_TOO_SMALL (1408)
 Invalid value in `qcsubi`.

MSK_RES_ERR_QCON_SUBI_TOO_LARGE (1409)
 Invalid value in `qcsubi`.

MSK_RES_ERR_QOBJ_UPPER_TRIANGLE (1415)
 An element in the upper triangle of Q^o is specified. Only elements in the lower triangle should be specified.

MSK_RES_ERR_QCON_UPPER_TRIANGLE (1417)
 An element in the upper triangle of a Q^k is specified. Only elements in the lower triangle should be specified.

MSK_RES_ERR_FIXED_BOUND_VALUES (1420)
 A fixed constraint/variable has been specified using the bound keys but the numerical value of the lower and upper bound is different.

MSK_RES_ERR_TOO_SMALL_A_TRUNCATION_VALUE (1421)
 A too small value for the A truncation value is specified.

MSK_RES_ERR_INVALID_OBJECTIVE_SENSE (1445)
 An invalid objective sense is specified.

MSK_RES_ERR_UNDEFINED_OBJECTIVE_SENSE (1446)
 The objective sense has not been specified before the optimization.

MSK_RES_ERR_Y_IS_UNDEFINED (1449)
 The solution item y is undefined.

MSK_RES_ERR_NAN_IN_DOUBLE_DATA (1450)
 An invalid floating point value was used in some double data.

MSK_RES_ERR_INF_IN_DOUBLE_DATA (1451)
 An infinite floating point value was used in some double data.

MSK_RES_ERR_NAN_IN_BLC (1461)
 l^c contains an invalid floating point value, i.e. a NaN.

MSK_RES_ERR_NAN_IN_BUC (1462)
 u^c contains an invalid floating point value, i.e. a NaN.

MSK_RES_ERR_INVALID_CFIX (1469)
 An invalid fixed term in the objective is specified.

MSK_RES_ERR_NAN_IN_C (1470)
 c contains an invalid floating point value, i.e. a NaN.

MSK_RES_ERR_NAN_IN_BLX (1471)
 l^x contains an invalid floating point value, i.e. a NaN.

MSK_RES_ERR_NAN_IN_BUX (1472)
 u^x contains an invalid floating point value, i.e. a NaN.

MSK_RES_ERR_INVALID_AIJ (1473)
 $a_{i,j}$ contains an invalid floating point value, i.e. a NaN or an infinite value.

MSK_RES_ERR_INVALID_CJ (1474)
 c_j contains an invalid floating point value, i.e. a NaN or an infinite value.

MSK_RES_ERR_SYM_MAT_INVALID (1480)
 A symmetric matrix contains an invalid floating point value, i.e. a NaN or an infinite value.

MSK_RES_ERR_SYM_MAT_HUGE (1482)
 A symmetric matrix contains a huge value in absolute size. The parameter `MSK_DPAR_DATA_SYM_MAT_TOL_HUGE` controls when an $e_{i,j}$ is considered huge.

MSK_RES_ERR_INV_PROBLEM (1500)
 Invalid problem type. Probably a nonconvex problem has been specified.

MSK_RES_ERR_MIXED_CONIC_AND_NL (1501)
 The problem contains nonlinear terms conic constraints. The requested operation cannot be applied to this type of problem.

MSK_RES_ERR_GLOBAL_INV_CONIC_PROBLEM (1503)
 The global optimizer can only be applied to problems without semidefinite variables.

MSK_RES_ERR_INV_OPTIMIZER (1550)
 An invalid optimizer has been chosen for the problem.

MSK_RES_ERR_MIO_NO_OPTIMIZER (1551)
 No optimizer is available for the current class of integer optimization problems.

MSK_RES_ERR_NO_OPTIMIZER_VAR_TYPE (1552)
 No optimizer is available for this class of optimization problems.

MSK_RES_ERR_FINAL_SOLUTION (1560)
 An error occurred during the solution finalization.

MSK_RES_ERR_FIRST (1570)
 Invalid `first`.

MSK_RES_ERR_LAST (1571)
 Invalid index `last`. A given index was out of expected range.

MSK_RES_ERR_SLICE_SIZE (1572)
 Invalid slice size specified.

MSK_RES_ERR_NEGATIVE_SURPLUS (1573)
 Negative surplus.

MSK_RES_ERR_NEGATIVE_APPEND (1578)
 Cannot append a negative number.

MSK_RES_ERR_POSTSOLVE (1580)
 An error occurred during the postsolve. Please contact **MOSEK** support.

MSK_RES_ERR_OVERFLOW (1590)
 A computation produced an overflow i.e. a very large number.

MSK_RES_ERR_NO_BASIS_SOL (1600)
 No basic solution is defined.

MSK_RES_ERR_BASIS_FACTOR (1610)
 The factorization of the basis is invalid.

MSK_RES_ERR_BASIS_SINGULAR (1615)
 The basis is singular and hence cannot be factored.

MSK_RES_ERR_FACTOR (1650)
 An error occurred while factorizing a matrix.

MSK_RES_ERR_FEASREPAIR_CANNOT_RELAX (1700)
 An optimization problem cannot be relaxed.

MSK_RES_ERR_FEASREPAIR_SOLVING_RELAXED (1701)
 The relaxed problem could not be solved to optimality. Please consult the log file for further details.

MSK_RES_ERR_FEASREPAIR_INCONSISTENT_BOUND (1702)
 The upper bound is less than the lower bound for a variable or a constraint. Please correct this before running the feasibility repair.

MSK_RES_ERR_REPAIR_INVALID_PROBLEM (1710)
 The feasibility repair does not support the specified problem type.

MSK_RES_ERR_REPAIR_OPTIMIZATION_FAILED (1711)
 Computation the optimal relaxation failed. The cause may have been numerical problems.

MSK_RES_ERR_NAME_MAX_LEN (1750)
 A name is longer than the buffer that is supposed to hold it.

MSK_RES_ERR_NAME_IS_NULL (1760)
 The name buffer is a NULL pointer.

MSK_RES_ERR_INVALID_COMPRESSION (1800)
 Invalid compression type.

MSK_RES_ERR_INVALID_IOMODE (1801)
 Invalid io mode.

MSK_RES_ERR_NO_PRIMAL_INFEAS_CER (2000)
 A certificate of primal infeasibility is not available.

MSK_RES_ERR_NO_DUAL_INFEAS_CER (2001)
 A certificate of infeasibility is not available.

MSK_RES_ERR_NO_SOLUTION_IN_CALLBACK (2500)
 The required solution is not available.

MSK_RES_ERR_INV_MARKI (2501)
 Invalid value in marki.

MSK_RES_ERR_INV_MARKJ (2502)
 Invalid value in markj.

MSK_RES_ERR_INV_NUMI (2503)
 Invalid numi.

MSK_RES_ERR_INV_NUMJ (2504)
 Invalid numj.

MSK_RES_ERR_TASK_INCOMPATIBLE (2560)
 The Task file is incompatible with this platform. This results from reading a file on a 32 bit platform generated on a 64 bit platform.

MSK_RES_ERR_TASK_INVALID (2561)
 The Task file is invalid.

MSK_RES_ERR_TASK_WRITE (2562)
 Failed to write the task file.

MSK_RES_ERR_LU_MAX_NUM_TRIES (2800)
 Could not compute the LU factors of the matrix within the maximum number of allowed tries.

MSK_RES_ERR_INVALID_UTF8 (2900)
 An invalid UTF8 string is encountered.

MSK_RES_ERR_INVALID_WCHAR (2901)
 An invalid wchar string is encountered.

MSK_RES_ERR_NO_DUAL_FOR_ITG_SOL (2950)
 No dual information is available for the integer solution.

MSK_RES_ERR_NO_SNX_FOR_BAS_SOL (2953)
 s_n^x is not available for the basis solution.

MSK_RES_ERR_INTERNAL (3000)
 An internal error occurred. Please report this problem.

MSK_RES_ERR_API_ARRAY_TOO_SMALL (3001)
 An input array was too short.

MSK_RES_ERR_API_CB_CONNECT (3002)
 Failed to connect a callback object.

MSK_RES_ERR_API_FATAL_ERROR (3005)
 An internal error occurred in the API. Please report this problem.

MSK_RES_ERR_API_INTERNAL (3999)
 An internal fatal error occurred in an interface function.

MSK_RES_ERR_SEN_FORMAT (3050)
 Syntax error in sensitivity analysis file.

MSK_RES_ERR_SEN_UNDEF_NAME (3051)
 An undefined name was encountered in the sensitivity analysis file.

MSK_RES_ERR_SEN_INDEX_RANGE (3052)
 Index out of range in the sensitivity analysis file.

MSK_RES_ERR_SEN_BOUND_INVALID_UP (3053)
 Analysis of upper bound requested for an index, where no upper bound exists.

MSK_RES_ERR_SEN_BOUND_INVALID_LO (3054)
 Analysis of lower bound requested for an index, where no lower bound exists.

MSK_RES_ERR_SEN_INDEX_INVALID (3055)
 Invalid range given in the sensitivity file.

MSK_RES_ERR_SEN_INVALID_REGEX (3056)
 Syntax error in regexp or regexp longer than 1024.

MSK_RES_ERR_SEN_SOLUTION_STATUS (3057)
 No optimal solution found to the original problem given for sensitivity analysis.

MSK_RES_ERR_SEN_NUMERICAL (3058)
 Numerical difficulties encountered performing the sensitivity analysis.

MSK_RES_ERR_SEN_UNHANDLED_PROBLEM_TYPE (3080)
 Sensitivity analysis cannot be performed for the specified problem. Sensitivity analysis is only possible for linear problems.

MSK_RES_ERR_UNB_STEP_SIZE (3100)
 A step size in an optimizer was unexpectedly unbounded. For instance, if the step-size becomes unbounded in phase 1 of the simplex algorithm then an error occurs. Normally this will happen only if the problem is badly formulated. Please contact **MOSEK** support if this error occurs.

MSK_RES_ERR_IDENTICAL_TASKS (3101)
 Some tasks related to this function call were identical. Unique tasks were expected.

MSK_RES_ERR_AD_INVALID_CODELIST (3102)
 The code list data was invalid.

MSK_RES_ERR_INTERNAL_TEST_FAILED (3500)
 An internal unit test function failed.

MSK_RES_ERR_XML_INVALID_PROBLEM_TYPE (3600)
 The problem type is not supported by the XML format.

MSK_RES_ERR_INVALIDAMPL_STUB (3700)
 Invalid AMPL stub.

MSK_RES_ERR_INT64_TO_INT32_CAST (3800)
 A 64 bit integer could not be cast to a 32 bit integer.

MSK_RES_ERR_SIZE_LICENSE_NUMCORES (3900)
 The computer contains more cpu cores than the license allows for.

MSK_RES_ERR_INFEAS_UNDEFINED (3910)
 The requested value is not defined for this solution type.

MSK_RES_ERR_NO_BARX_FOR_SOLUTION (3915)
 There is no \bar{X} available for the solution specified. In particular note there are no \bar{X} defined for the basic and integer solutions.

MSK_RES_ERR_NO_BARS_FOR_SOLUTION (3916)
 There is no \bar{s} available for the solution specified. In particular note there are no \bar{s} defined for the basic and integer solutions.

MSK_RES_ERR_BAR_VAR_DIM (3920)
 The dimension of a symmetric matrix variable has to be greater than 0.

MSK_RES_ERR_SYM_MAT_INVALID_ROW_INDEX (3940)
 A row index specified for sparse symmetric matrix is invalid.

MSK_RES_ERR_SYM_MAT_INVALID_COL_INDEX (3941)
 A column index specified for sparse symmetric matrix is invalid.

MSK_RES_ERR_SYM_MAT_NOT_LOWER_TRINGULAR (3942)
 Only the lower triangular part of sparse symmetric matrix should be specified.

MSK_RES_ERR_SYM_MAT_INVALID_VALUE (3943)
 The numerical value specified in a sparse symmetric matrix is not a floating point value.

MSK_RES_ERR_SYM_MAT_DUPLICATE (3944)
 A value in a symmetric matrix as been specified more than once.

MSK_RES_ERR_INVALID_SYM_MAT_DIM (3950)
 A sparse symmetric matrix of invalid dimension is specified.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_SYM_MAT (4000)
 The file format does not support a problem with symmetric matrix variables.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_CFIX (4001)
 The file format does not support a problem with nonzero fixed term in c.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_RANGED_CONSTRAINTS (4002)
 The file format does not support a problem with ranged constraints.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_FREE_CONSTRAINTS (4003)
 The file format does not support a problem with free constraints.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_CONES (4005)
 The file format does not support a problem with conic constraints.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_QUADRATIC_TERMS (4006)
 The file format does not support a problem with quadratic terms.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_NONLINEAR (4010)
 The file format does not support a problem with nonlinear terms.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_DISJUNCTIVE_CONSTRAINTS (4011)
 The file format does not support a problem with disjunctive constraints.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_AFFINE_CONIC_CONSTRAINTS (4012)
 The file format does not support a problem with affine conic constraints.

MSK_RES_ERR_DUPLICATE_CONSTRAINT_NAMES (4500)
 Two constraint names are identical.

MSK_RES_ERR_DUPLICATE_VARIABLE_NAMES (4501)
 Two variable names are identical.

MSK_RES_ERR_DUPLICATE_BARVARIABLE_NAMES (4502)
 Two barvariable names are identical.

MSK_RES_ERR_DUPLICATE_CONE_NAMES (4503)
 Two cone names are identical.

MSK_RES_ERR_DUPLICATE_DOMAIN_NAMES (4504)
 Two domain names are identical.

MSK_RES_ERR_DUPLICATE_DJC_NAMES (4505)
 Two disjunctive constraint names are identical.

MSK_RES_ERR_NON_UNIQUE_ARRAY (5000)
 An array does not contain unique elements.

MSK_RES_ERR_ARGUMENT_IS_TOO_SMALL (5004)
 The value of a function argument is too small.

MSK_RES_ERR_ARGUMENT_IS_TOO_LARGE (5005)
 The value of a function argument is too large.

MSK_RES_ERR_MIO_INTERNAL (5010)
 A fatal error occurred in the mixed integer optimizer. Please contact **MOSEK** support.

MSK_RES_ERR_INVALID_PROBLEM_TYPE (6000)
 An invalid problem type.

MSK_RES_ERR_UNHANDLED_SOLUTION_STATUS (6010)
 Unhandled solution status.

MSK_RES_ERR_UPPER_TRIANGLE (6020)
 An element in the upper triangle of a lower triangular matrix is specified.

MSK_RES_ERR_LAU_SINGULAR_MATRIX (7000)
 A matrix is singular.

MSK_RES_ERR_LAU_NOT_POSITIVE_DEFINITE (7001)
 A matrix is not positive definite.

MSK_RES_ERR_LAU_INVALID_LOWER_TRIANGULAR_MATRIX (7002)
 An invalid lower triangular matrix.

MSK_RES_ERR_LAU_UNKNOWN (7005)
 An unknown error.

MSK_RES_ERR_LAU_ARG_M (7010)
 Invalid argument m.

MSK_RES_ERR_LAU_ARG_N (7011)
 Invalid argument n.

MSK_RES_ERR_LAU_ARG_K (7012)
 Invalid argument k.

MSK_RES_ERR_LAU_ARG_TRANSA (7015)
 Invalid argument transa.

MSK_RES_ERR_LAU_ARG_TRANSB (7016)
 Invalid argument transb.

MSK_RES_ERR_LAU_ARG_UPLO (7017)
 Invalid argument uplo.

MSK_RES_ERR_LAU_ARG_TRANS (7018)
 Invalid argument trans.

MSK_RES_ERR_LAU_INVALID_SPARSE_SYMMETRIC_MATRIX (7019)
 An invalid sparse symmetric matrix is specified. Note only the lower triangular part with no duplicates is specified.

MSK_RES_ERR_CBF_PARSE (7100)
 An error occurred while parsing an CBF file.

MSK_RES_ERR_CBF_OBJ_SENSE (7101)
 An invalid objective sense is specified.

MSK_RES_ERR_CBF_NO_VARIABLES (7102)
 No variables are specified.

MSK_RES_ERR_CBF_TOO_MANY_CONSTRAINTS (7103)
 Too many constraints specified.

MSK_RES_ERR_CBF_TOO_MANY_VARIABLES (7104)
 Too many variables specified.

MSK_RES_ERR_CBF_NO_VERSION_SPECIFIED (7105)
 No version specified.

MSK_RES_ERR_CBF_SYNTAX (7106)
 Invalid syntax.

MSK_RES_ERR_CBF_DUPLICATE_OBJ (7107)
 Duplicate OBJ keyword.

MSK_RES_ERR_CBF_DUPLICATE_CON (7108)
 Duplicate CON keyword.

MSK_RES_ERR_CBF_DUPLICATE_VAR (7110)
 Duplicate VAR keyword.

MSK_RES_ERR_CBF_DUPLICATE_INT (7111)
 Duplicate INT keyword.

MSK_RES_ERR_CBF_INVALID_VAR_TYPE (7112)
 Invalid variable type.

MSK_RES_ERR_CBF_INVALID_CON_TYPE (7113)
 Invalid constraint type.

MSK_RES_ERR_CBF_INVALID_DOMAIN_DIMENSION (7114)
 Invalid domain dimension.

MSK_RES_ERR_CBF_DUPLICATE_OBJCOORD (7115)
 Duplicate index in OBJCOORD.

MSK_RES_ERR_CBF_DUPLICATE_BCOORD (7116)
 Duplicate index in BCOORD.

MSK_RES_ERR_CBF_DUPLICATE_ACOORD (7117)
 Duplicate index in ACOORD.

MSK_RES_ERR_CBF_TOO_FEW_VARIABLES (7118)
 Too few variables defined.

MSK_RES_ERR_CBF_TOO_FEW_CONSTRAINTS (7119)
 Too few constraints defined.

MSK_RES_ERR_CBF_TOO_FEW_INTS (7120)
Too few ints are specified.

MSK_RES_ERR_CBF_TOO_MANY_INTS (7121)
Too many ints are specified.

MSK_RES_ERR_CBF_INVALID_INT_INDEX (7122)
Invalid INT index.

MSK_RES_ERR_CBF_UNSUPPORTED (7123)
Unsupported feature is present.

MSK_RES_ERR_CBF_DUPLICATE_PSDVAR (7124)
Duplicate PSDVAR keyword.

MSK_RES_ERR_CBF_INVALID_PSDVAR_DIMENSION (7125)
Invalid PSDVAR dimension.

MSK_RES_ERR_CBF_TOO_FEW_PSDVAR (7126)
Too few variables defined.

MSK_RES_ERR_CBF_INVALID_EXP_DIMENSION (7127)
Invalid dimension of a exponential cone.

MSK_RES_ERR_CBF_DUPLICATE_POW_CONES (7130)
Multiple POWCONES specified.

MSK_RES_ERR_CBF_DUPLICATE_POW_STAR_CONES (7131)
Multiple POW*CONES specified.

MSK_RES_ERR_CBF_INVALID_POWER (7132)
Invalid power specified.

MSK_RES_ERR_CBF_POWER_CONE_IS_TOO_LONG (7133)
Power cone is too long.

MSK_RES_ERR_CBF_INVALID_POWER_CONE_INDEX (7134)
Invalid power cone index.

MSK_RES_ERR_CBF_INVALID_POWER_STAR_CONE_INDEX (7135)
Invalid power star cone index.

MSK_RES_ERR_CBF_UNHANDLED_POWER_CONE_TYPE (7136)
An unhandled power cone type.

MSK_RES_ERR_CBF_UNHANDLED_POWER_STAR_CONE_TYPE (7137)
An unhandled power star cone type.

MSK_RES_ERR_CBF_POWER_CONE_MISMATCH (7138)
The power cone does not match with it definition.

MSK_RES_ERR_CBF_POWER_STAR_CONE_MISMATCH (7139)
The power star cone does not match with it definition.

MSK_RES_ERR_CBF_INVALID_NUMBER_OF_CONES (7140)
Invalid number of cones.

MSK_RES_ERR_CBF_INVALID_DIMENSION_OF_CONES (7141)
Invalid number of cones.

MSK_RES_ERR_CBF_INVALID_NUM_PSDCON (7200)
Invalid number of PSDCON.

MSK_RES_ERR_CBF_DUPLICATE_PSDCON (7201)
Duplicate CON keyword.

MSK_RES_ERR_CBF_INVALID_DIMENSION_OF_PSDCON (7202)
Invalid PSDCON dimension.

MSK_RES_ERR_CBF_INVALID_PSDCON_INDEX (7203)
Invalid PSDCON index.

MSK_RES_ERR_CBF_INVALID_PSDCON_VARIABLE_INDEX (7204)
Invalid PSDCON index.

MSK_RES_ERR_CBF_INVALID_PSDCON_BLOCK_INDEX (7205)
Invalid PSDCON index.

MSK_RES_ERR_CBF_UNSUPPORTED_CHANGE (7210)
The CHANGE section is not supported.

MSK_RES_ERR_MIO_INVALID_ROOT_OPTIMIZER (7700)
An invalid root optimizer was selected for the problem type.

MSK_RES_ERR_MIO_INVALID_NODE_OPTIMIZER (7701)
An invalid node optimizer was selected for the problem type.

MSK_RES_ERR_MPS_WRITE_CPLEX_INVALID_CONE_TYPE (7750)
 An invalid cone type occurs when writing a CPLEX formatted MPS file.

MSK_RES_ERR_TOCONIC_CONSTR_Q_NOT_PSD (7800)
 The matrix defining the quadratic part of constraint is not positive semidefinite.

MSK_RES_ERR_TOCONIC_CONSTRAINT_FX (7801)
 The quadratic constraint is an equality, thus not convex.

MSK_RES_ERR_TOCONIC_CONSTRAINT_RA (7802)
 The quadratic constraint has finite lower and upper bound, and therefore it is not convex.

MSK_RES_ERR_TOCONIC_CONSTR_NOT_CONIC (7803)
 The constraint is not conic representable.

MSK_RES_ERR_TOCONIC_OBJECTIVE_NOT_PSD (7804)
 The matrix defining the quadratic part of the objective function is not positive semidefinite.

MSK_RES_ERR_SERVER_CONNECT (8000)
 Failed to connect to remote solver server. The server string or the port string were invalid, or the server did not accept connection.

MSK_RES_ERR_SERVER_PROTOCOL (8001)
 Unexpected message or data from solver server.

MSK_RES_ERR_SERVER_STATUS (8002)
 Server returned non-ok HTTP status code

MSK_RES_ERR_SERVER_TOKEN (8003)
 The job ID specified is incorrect or invalid

MSK_RES_ERR_SERVER_ADDRESS (8004)
 Invalid address string

MSK_RES_ERR_SERVER_CERTIFICATE (8005)
 Invalid TLS certificate format or path

MSK_RES_ERR_SERVER_TLS_CLIENT (8006)
 Failed to create TLS client

MSK_RES_ERR_SERVER_ACCESS_TOKEN (8007)
 Invalid access token

MSK_RES_ERR_SERVER_PROBLEM_SIZE (8008)
 The size of the problem exceeds the dimensions permitted by the instance of the OptServer where it was run.

MSK_RES_ERR_DUPLICATE_FIJ (20100)
 An element in the F matrix is specified twice.

MSK_RES_ERR_INVALID_FIJ (20101)
 $f_{i,j}$ contains an invalid floating point value, i.e. a NaN or an infinite value.

MSK_RES_ERR_HUGE_FIJ (20102)
 A numerically huge value is specified for an $f_{i,j}$ element in F . The parameter [MSK_DPAR_DATA_TOL_AIJ_HUGE](#) controls when an $f_{i,j}$ is considered huge.

MSK_RES_ERR_INVALID_G (20103)
 g contains an invalid floating point value, i.e. a NaN or an infinite value.

MSK_RES_ERR_INVALID_B (20150)
 b contains an invalid floating point value, i.e. a NaN or an infinite value.

MSK_RES_ERR_DOMAIN_INVALID_INDEX (20400)
 A domain index is invalid.

MSK_RES_ERR_DOMAIN_DIMENSION (20401)
 A domain dimension is invalid.

MSK_RES_ERR_DOMAIN_DIMENSION_PSD (20402)
 A PSD domain dimension is invalid.

MSK_RES_ERR_NOT_POWER_DOMAIN (20403)
 The function is only applicable to primal and dual power cone domains.

MSK_RES_ERR_DOMAIN_POWER_INVALID_ALPHA (20404)
 Alpha contains an invalid floating point value, i.e. a NaN or an infinite value.

MSK_RES_ERR_DOMAIN_POWER_NEGATIVE_ALPHA (20405)
 Alpha contains a negative value or zero.

MSK_RES_ERR_DOMAIN_POWER_NLEFT (20406)
 The value of n_{left} is not in $[1, n - 1]$ where n is the dimension.

MSK_RES_ERR_AFE_INVALID_INDEX (20500)
 An affine expression index is invalid.

MSK_RES_ERR_ACC_INVALID_INDEX (20600)
 A affine conic constraint index is invalid.

MSK_RES_ERR_ACC_INVALID_ENTRY_INDEX (20601)
 The index of an element in an affine conic constraint is invalid.

MSK_RES_ERR_ACC_AFE_DOMAIN_MISMATCH (20602)
 There is a mismatch between between the number of affine expressions and total dimension of the domain(s).

MSK_RES_ERR_DJC_INVALID_INDEX (20700)
 A disjunctive constraint index is invalid.

MSK_RES_ERR_DJC_UNSUPPORTED_DOMAIN_TYPE (20701)
 An unsupported domain type has been used in a disjunctive constraint.

MSK_RES_ERR_DJC_AFE_DOMAIN_MISMATCH (20702)
 There is a mismatch between the number of affine expressions and total dimension of the domain(s).

MSK_RES_ERR_DJC_INVALID_TERM_SIZE (20703)
 A termize is invalid.

MSK_RES_ERR_DJC_DOMAIN_TERMSIZE_MISMATCH (20704)
 There is a mismatch between the number of domains and the term sizes.

MSK_RES_ERR_DJC_TOTAL_NUM_TERMS_MISMATCH (20705)
 There total number of terms in all domains does not match.

MSK_RES_ERR_UNDEF_SOLUTION (22000)
MOSEK has the following solution types:

- an interior-point solution,
- a basic solution,
- and an integer solution.

Each optimizer may set one or more of these solutions; e.g by default a successful optimization with the interior-point optimizer defines the interior-point solution and, for linear problems, also the basic solution. This error occurs when asking for a solution or for information about a solution that is not defined.

MSK_RES_ERR_NO_DOTY (22010)
 No doty is available

11.4 Constants

11.4.1 Basis identification

MSK_BI_NEVER
 Never do basis identification.

MSK_BI_ALWAYS
 Basis identification is always performed even if the interior-point optimizer terminates abnormally.

MSK_BI_NO_ERROR
 Basis identification is performed if the interior-point optimizer terminates without an error.

MSK_BI_IF_FEASIBLE
 Basis identification is not performed if the interior-point optimizer terminates with a problem status saying that the problem is primal or dual infeasible.

MSK_BI_RESERVED
 Not currently in use.

11.4.2 Bound keys

MSK_BK_LO

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

MSK_BK_UP

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

MSK_BK_FX

The constraint or variable is fixed.

MSK_BK_FR

The constraint or variable is free.

MSK_BK_RA

The constraint or variable is ranged.

11.4.3 Mark

MSK_MARK_LO

The lower bound is selected for sensitivity analysis.

MSK_MARK_UP

The upper bound is selected for sensitivity analysis.

11.4.4 Degeneracy strategies

MSK_SIM_DEGEN_NONE

The simplex optimizer should use no degeneration strategy.

MSK_SIM_DEGEN_FREE

The simplex optimizer chooses the degeneration strategy.

MSK_SIM_DEGEN_AGGRESSIVE

The simplex optimizer should use an aggressive degeneration strategy.

MSK_SIM_DEGEN_MODERATE

The simplex optimizer should use a moderate degeneration strategy.

MSK_SIM_DEGEN_MINIMUM

The simplex optimizer should use a minimum degeneration strategy.

11.4.5 Transposed matrix.

MSK_TRANSPOSE_NO

No transpose is applied.

MSK_TRANSPOSE_YES

A transpose is applied.

11.4.6 Triangular part of a symmetric matrix.

MSK_UPLO_LO

Lower part.

MSK_UPLO_UP

Upper part.

11.4.7 Problem reformulation.

MSK_SIM_REFORMULATION_ON

Allow the simplex optimizer to reformulate the problem.

MSK_SIM_REFORMULATION_OFF

Disallow the simplex optimizer to reformulate the problem.

MSK_SIM_REFORMULATION_FREE

The simplex optimizer can choose freely.

MSK_SIM_REFORMULATION_AGGRESSIVE

The simplex optimizer should use an aggressive reformulation strategy.

11.4.8 Exploit duplicate columns.

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.

11.4.9 Hot-start type employed by the simplex optimizer

MSK_SIM_HOTSTART_NONE

The simplex optimizer performs a coldstart.

MSK_SIM_HOTSTART_FREE

The simplex optimizer chooses the hot-start type.

MSK_SIM_HOTSTART_STATUS_KEYS

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

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

MSK_INTPNT_HOTSTART_NONE

The interior-point optimizer performs a coldstart.

MSK_INTPNT_HOTSTART_PRIMAL

The interior-point optimizer exploits the primal solution only.

MSK_INTPNT_HOTSTART_DUAL

The interior-point optimizer exploits the dual solution only.

MSK_INTPNT_HOTSTART_PRIMAL_DUAL

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

11.4.11 Solution purification employed optimizer.

MSK_PURIFY_NONE

The optimizer performs no solution purification.

MSK_PURIFY_PRIMAL

The optimizer purifies the primal solution.

MSK_PURIFY_DUAL

The optimizer purifies the dual solution.

MSK_PURIFY_PRIMAL_DUAL

The optimizer purifies both the primal and dual solution.

MSK_PURIFY_AUTO

TBD

11.4.12 Progress callback codes

MSK_CALLBACK_BEGIN_BI

The basis identification procedure has been started.

MSK_CALLBACK_BEGIN_CONIC

The callback function is called when the conic optimizer is started.

MSK_CALLBACK_BEGIN_DUAL_BI

The callback function is called from within the basis identification procedure when the dual phase is started.

MSK_CALLBACK_BEGIN_DUAL_SENSITIVITY

Dual sensitivity analysis is started.

MSK_CALLBACK_BEGIN_DUAL_SETUP_BI

The callback function is called when the dual BI phase is started.

MSK_CALLBACK_BEGIN_DUAL_SIMPLEX

The callback function is called when the dual simplex optimizer started.

MSK_CALLBACK_BEGIN_DUAL_SIMPLEX_BI
The callback function is called from within the basis identification procedure when the dual simplex clean-up phase is started.

MSK_CALLBACK_BEGIN_INFEAS_ANA
The callback function is called when the infeasibility analyzer is started.

MSK_CALLBACK_BEGIN_INTPNT
The callback function is called when the interior-point optimizer is started.

MSK_CALLBACK_BEGIN_LICENSE_WAIT
Begin waiting for license.

MSK_CALLBACK_BEGIN_MIO
The callback function is called when the mixed-integer optimizer is started.

MSK_CALLBACK_BEGIN_OPTIMIZER
The callback function is called when the optimizer is started.

MSK_CALLBACK_BEGIN_PRESOLVE
The callback function is called when the presolve is started.

MSK_CALLBACK_BEGIN_PRIMAL_BI
The callback function is called from within the basis identification procedure when the primal phase is started.

MSK_CALLBACK_BEGIN_PRIMAL_REPAIR
Begin primal feasibility repair.

MSK_CALLBACK_BEGIN_PRIMAL_SENSITIVITY
Primal sensitivity analysis is started.

MSK_CALLBACK_BEGIN_PRIMAL_SETUP_BI
The callback function is called when the primal BI setup is started.

MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX
The callback function is called when the primal simplex optimizer is started.

MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX_BI
The callback function is called from within the basis identification procedure when the primal simplex clean-up phase is started.

MSK_CALLBACK_BEGIN_QCQO_REFORMULATE
Begin QCQO reformulation.

MSK_CALLBACK_BEGIN_READ
MOSEK has started reading a problem file.

MSK_CALLBACK_BEGIN_ROOT_CUTGEN
The callback function is called when root cut generation is started.

MSK_CALLBACK_BEGIN_SIMPLEX
The callback function is called when the simplex optimizer is started.

MSK_CALLBACK_BEGIN_SIMPLEX_BI
The callback function is called from within the basis identification procedure when the simplex clean-up phase is started.

MSK_CALLBACK_BEGIN_SOLVE_ROOT_RELAX
The callback function is called when solution of root relaxation is started.

MSK_CALLBACK_BEGIN_TO_CONIC
Begin conic reformulation.

MSK_CALLBACK_BEGIN_WRITE
MOSEK has started writing a problem file.

MSK_CALLBACK_CONIC
The callback function is called from within the conic optimizer after the information database has been updated.

MSK_CALLBACK_DUAL_SIMPLEX
The callback function is called from within the dual simplex optimizer.

MSK_CALLBACK_END_BI
The callback function is called when the basis identification procedure is terminated.

MSK_CALLBACK_END_CONIC
The callback function is called when the conic optimizer is terminated.

MSK_CALLBACK_END_DUAL_BI
The callback function is called from within the basis identification procedure when the dual phase is terminated.

MSK_CALLBACK_END_DUAL_SENSITIVITY
Dual sensitivity analysis is terminated.

MSK_CALLBACK_END_DUAL_SETUP_BI
The callback function is called when the dual BI phase is terminated.

MSK_CALLBACK_END_DUAL_SIMPLEX
The callback function is called when the dual simplex optimizer is terminated.

MSK_CALLBACK_END_DUAL_SIMPLEX_BI
The callback function is called from within the basis identification procedure when the dual clean-up phase is terminated.

MSK_CALLBACK_END_INFEAS_ANA
The callback function is called when the infeasibility analyzer is terminated.

MSK_CALLBACK_END_INTPNT
The callback function is called when the interior-point optimizer is terminated.

MSK_CALLBACK_END_LICENSE_WAIT
End waiting for license.

MSK_CALLBACK_END_MIO
The callback function is called when the mixed-integer optimizer is terminated.

MSK_CALLBACK_END_OPTIMIZER
The callback function is called when the optimizer is terminated.

MSK_CALLBACK_END_PRESOLVE
The callback function is called when the presolve is completed.

MSK_CALLBACK_END_PRIMAL_BI
The callback function is called from within the basis identification procedure when the primal phase is terminated.

MSK_CALLBACK_END_PRIMAL_REPAIR
End primal feasibility repair.

MSK_CALLBACK_END_PRIMAL_SENSITIVITY
Primal sensitivity analysis is terminated.

MSK_CALLBACK_END_PRIMAL_SETUP_BI
The callback function is called when the primal BI setup is terminated.

MSK_CALLBACK_END_PRIMAL_SIMPLEX
The callback function is called when the primal simplex optimizer is terminated.

MSK_CALLBACK_END_PRIMAL_SIMPLEX_BI
The callback function is called from within the basis identification procedure when the primal clean-up phase is terminated.

MSK_CALLBACK_END_QCQO_REFORMULATE
End QCQO reformulation.

MSK_CALLBACK_END_READ
MOSEK has finished reading a problem file.

MSK_CALLBACK_END_ROOT_CUTGEN
The callback function is called when root cut generation is terminated.

MSK_CALLBACK_END_SIMPLEX
The callback function is called when the simplex optimizer is terminated.

MSK_CALLBACK_END_SIMPLEX_BI
The callback function is called from within the basis identification procedure when the simplex clean-up phase is terminated.

MSK_CALLBACK_END_SOLVE_ROOT_RELAX
The callback function is called when solution of root relaxation is terminated.

MSK_CALLBACK_END_TO_CONIC
End conic reformulation.

MSK_CALLBACK_END_WRITE
MOSEK has finished writing a problem file.

MSK_CALLBACK_IM_BI
The callback function is called from within the basis identification procedure at an intermediate point.

MSK_CALLBACK_IM_CONIC
The callback function is called at an intermediate stage within the conic optimizer where the information database has not been updated.

MSK_CALLBACK_IM_DUAL_BI
The callback function is called from within the basis identification procedure at an intermediate point in the dual phase.

MSK_CALLBACK_IM_DUAL_SENSIVITY
The callback function is called at an intermediate stage of the dual sensitivity analysis.

MSK_CALLBACK_IM_DUAL_SIMPLEX
The callback function is called at an intermediate point in the dual simplex optimizer.

MSK_CALLBACK_IM_INTPNT
The callback function is called at an intermediate stage within the interior-point optimizer where the information database has not been updated.

MSK_CALLBACK_IM_LICENSE_WAIT
MOSEK is waiting for a license.

MSK_CALLBACK_IM_LU
The callback function is called from within the LU factorization procedure at an intermediate point.

MSK_CALLBACK_IM_MIO
The callback function is called at an intermediate point in the mixed-integer optimizer.

MSK_CALLBACK_IM_MIO_DUAL_SIMPLEX
The callback function is called at an intermediate point in the mixed-integer optimizer while running the dual simplex optimizer.

MSK_CALLBACK_IM_MIO_INTPNT
The callback function is called at an intermediate point in the mixed-integer optimizer while running the interior-point optimizer.

MSK_CALLBACK_IM_MIO_PRIMAL_SIMPLEX
The callback function is called at an intermediate point in the mixed-integer optimizer while running the primal simplex optimizer.

MSK_CALLBACK_IM_ORDER
The callback function is called from within the matrix ordering procedure at an intermediate point.

MSK_CALLBACK_IM_PRESOLVE
The callback function is called from within the presolve procedure at an intermediate stage.

MSK_CALLBACK_IM_PRIMAL_BI
The callback function is called from within the basis identification procedure at an intermediate point in the primal phase.

MSK_CALLBACK_IM_PRIMAL_SENSIVITY
The callback function is called at an intermediate stage of the primal sensitivity analysis.

MSK_CALLBACK_IM_PRIMAL_SIMPLEX
The callback function is called at an intermediate point in the primal simplex optimizer.

MSK_CALLBACK_IM_QO_REFORMULATE
The callback function is called at an intermediate stage of the conic quadratic reformulation.

MSK_CALLBACK_IM_READ
Intermediate stage in reading.

MSK_CALLBACK_IM_ROOT_CUTGEN
The callback is called from within root cut generation at an intermediate stage.

MSK_CALLBACK_IM_SIMPLEX
The callback function is called from within the simplex optimizer at an intermediate point.

MSK_CALLBACK_IM_SIMPLEX_BI
The callback function is called from within the basis identification procedure at an intermediate point in the simplex clean-up phase. The frequency of the callbacks is controlled by the *MSK_IPAR_LOG_SIM_FREQ* parameter.

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

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

MSK_CALLBACK_PRIMAL_SIMPLEX
The callback function is called from within the primal simplex optimizer.

MSK_CALLBACK_READ_OPF
The callback function is called from the OPF reader.

MSK_CALLBACK_READ_OPF_SECTION

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

MSK_CALLBACK_SOLVING_REMOTE

The callback function is called while the task is being solved on a remote server.

MSK_CALLBACK_UPDATE_DUAL_BI

The callback function is called from within the basis identification procedure at an intermediate point in the dual phase.

MSK_CALLBACK_UPDATE_DUAL_SIMPLEX

The callback function is called in the dual simplex optimizer.

MSK_CALLBACK_UPDATE_DUAL_SIMPLEX_BI

The callback function is called from within the basis identification procedure at an intermediate point in the dual simplex clean-up phase. The frequency of the callbacks is controlled by the *MSK_IPAR_LOG_SIM_FREQ* parameter.

MSK_CALLBACK_UPDATE_PRESOLVE

The callback function is called from within the presolve procedure.

MSK_CALLBACK_UPDATE_PRIMAL_BI

The callback function is called from within the basis identification procedure at an intermediate point in the primal phase.

MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX

The callback function is called in the primal simplex optimizer.

MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX_BI

The callback function is called from within the basis identification procedure at an intermediate point in the primal simplex clean-up phase. The frequency of the callbacks is controlled by the *MSK_IPAR_LOG_SIM_FREQ* parameter.

MSK_CALLBACK_UPDATE_SIMPLEX

The callback function is called from simplex optimizer.

MSK_CALLBACK_WRITE_OPF

The callback function is called from the OPF writer.

11.4.13 Types of convexity checks.

MSK_CHECK_CONVEXITY_NONE

No convexity check.

MSK_CHECK_CONVEXITY_SIMPLE

Perform simple and fast convexity check.

MSK_CHECK_CONVEXITY_FULL

Perform a full convexity check.

11.4.14 Compression types

MSK_COMPRESS_NONE

No compression is used.

MSK_COMPRESS_FREE

The type of compression used is chosen automatically.

MSK_COMPRESS_GZIP

The type of compression used is gzip compatible.

MSK_COMPRESS_ZSTD

The type of compression used is zstd compatible.

11.4.15 Cone types

MSK_CT_QUAD

The cone is a quadratic cone.

MSK_CT_RQUAD

The cone is a rotated quadratic cone.

MSK_CT_PEXP

A primal exponential cone.

MSK_CT_DEXP

A dual exponential cone.

MSK_CT_PPOW

A primal power cone.

MSK_CT_DPOW

A dual power cone.

MSK_CT_ZERO

The zero cone.

11.4.16 Cone types

MSK_DOMAIN_R

R.

MSK_DOMAIN_RZERO

The zero vector.

MSK_DOMAIN_RPLUS

The positive orthant.

MSK_DOMAIN_RMINUS

The negative orthant.

MSK_DOMAIN_QUADRATIC_CONE

The quadratic cone.

MSK_DOMAIN_RQUADRATIC_CONE

The rotated quadratic cone.

MSK_DOMAIN_PRIMAL_EXP_CONE

The primal exponential cone.

MSK_DOMAIN_DUAL_EXP_CONE

The dual exponential cone.

MSK_DOMAIN_PRIMAL_POWER_CONE

The primal power cone.

MSK_DOMAIN_DUAL_POWER_CONE

The dual power cone.

MSK_DOMAIN_PRIMAL_GEO_MEAN_CONE

The primal geometric mean cone.

MSK_DOMAIN_DUAL_GEO_MEAN_CONE

The dual geometric mean cone.

MSK_DOMAIN_SVEC_PSD_CONE

The vectorized positive semidefinite cone.

11.4.17 Name types

MSK_NAME_TYPE_GEN

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

MSK_NAME_TYPE_MPS

MPS type names.

MSK_NAME_TYPE_LP

LP type names.

11.4.18 Cone types

MSK_SYMMAT_TYPE_SPARSE

Sparse symmetric matrix.

11.4.19 Data format types

MSK_DATA_FORMAT_EXTENSION

The file extension is used to determine the data file format.

MSK_DATA_FORMAT_MPS

The data file is MPS formatted.

MSK_DATA_FORMAT_LP

The data file is LP formatted.

MSK_DATA_FORMAT_OP

The data file is an optimization problem formatted file.

MSK_DATA_FORMAT_FREE_MPS

The data a free MPS formatted file.

MSK_DATA_FORMAT_TASK

Generic task dump file.

MSK_DATA_FORMAT_PTF

(P)retty (T)ext (F)format.

MSK_DATA_FORMAT_CB

Conic benchmark format,

MSK_DATA_FORMAT_JSON_TASK

JSON based task format.

11.4.20 Data format types

MSK_SOL_FORMAT_EXTENSION

The file extension is used to determine the data file format.

MSK_SOL_FORMAT_B

Simple binary format

MSK_SOL_FORMAT_TASK

Tar based format.

MSK_SOL_FORMAT_JSON_TASK

JSON based format.

11.4.21 Double information items

MSK_DINF_ANA_PRO_SCALARIZED_CONSTRAINT_MATRIX_DENSITY

Density percentage of the scalarized constraint matrix.

MSK_DINF_BI_CLEAN_DUAL_TIME

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

MSK_DINF_BI_CLEAN_PRIMAL_TIME

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

MSK_DINF_BI_CLEAN_TIME

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

MSK_DINF_BI_DUAL_TIME

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

MSK_DINF_BI_PRIMAL_TIME

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

MSK_DINF_BI_TIME

Time spent within the basis identification procedure since its invocation.

MSK_DINF_INTPNT_DUAL_FEAS

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

MSK_DINF_INTPNT_DUAL_OBJ

Dual objective value reported by the interior-point optimizer.

MSK_DINF_INTPNT_FACTOR_NUM_FLOPS

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

MSK_DINF_INTPNT_OPT_STATUS

A measure of optimality of the solution. It should converge to +1 if the problem has a primal-dual optimal solution, and converge to -1 if the problem is (strictly) primal or dual infeasible. If the measure converges to another constant, or fails to settle, the problem is usually ill-posed.

MSK_DINF_INTPNT_ORDER_TIME

Order time (in seconds).

MSK_DINF_INTPNT_PRIMAL_FEAS

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

MSK_DINF_INTPNT_PRIMAL_OBJ

Primal objective value reported by the interior-point optimizer.

MSK_DINF_INTPNT_TIME

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

MSK_DINF_MIO_CLIQUÉ_SEPARATION_TIME

Separation time for clique cuts.

MSK_DINF_MIO_CMIR_SEPARATION_TIME

Separation time for CMIR cuts.

MSK_DINF_MIO_CONSTRUCT_SOLUTION_OBJ

If **MOSEK** has successfully constructed an integer feasible solution, then this item contains the optimal objective value corresponding to the feasible solution.

MSK_DINF_MIO_DUAL_BOUND_AFTER_PRESOLVE

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

MSK_DINF_MIO_GMI_SEPARATION_TIME

Separation time for GMI cuts.

MSK_DINF_MIO_IMPLIED_BOUND_TIME

Separation time for implied bound cuts.

MSK_DINF_MIO_INITIAL_FEASIBLE_SOLUTION_OBJ

If the user provided solution was found to be feasible this information item contains its objective value.

MSK_DINF_MIO_KNAPSACK_COVER_SEPARATION_TIME

Separation time for knapsack cover.

MSK_DINF_MIO_LIPRO_SEPARATION_TIME

Separation time for lift-and-project cuts.

MSK_DINF_MIO_OBJ_ABS_GAP

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

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

Otherwise it has the value -1.0.

MSK_DINF_MIO_OBJ_BOUND

The best known bound on the objective function. This value is undefined until at least one relaxation has been solved: To see if this is the case check that *MSK_IINF_MIO_NUM_RELAX* is strictly positive.

MSK_DINF_MIO_OBJ_INT

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

MSK_DINF_MIO_OBJ_REL_GAP

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

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

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

`MSK_DINF_MIO_PROBING_TIME`
Total time for probing.

`MSK_DINF_MIO_ROOT_CUTGEN_TIME`
Total time for cut generation.

`MSK_DINF_MIO_ROOT_OPTIMIZER_TIME`
Time spent in the continuous optimizer while processing the root node relaxation.

`MSK_DINF_MIO_ROOT_PRESOLVE_TIME`
Time spent presolving the problem at the root node.

`MSK_DINF_MIO_ROOT_TIME`
Time spent processing the root node.

`MSK_DINF_MIO_TIME`
Time spent in the mixed-integer optimizer.

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

`MSK_DINF_OPTIMIZER_TIME`
Total time spent in the optimizer since it was invoked.

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

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

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

`MSK_DINF_PRESOLVE_TOTAL_PRIMAL_PERTURBATION`
Total perturbation of the bounds of the primal problem.

`MSK_DINF_PRIMAL_REPAIR_PENALTY_OBJ`
The optimal objective value of the penalty function.

`MSK_DINF_QCQO_REFORMULATE_MAX_PERTURBATION`
Maximum absolute diagonal perturbation occurring during the QCQO reformulation.

`MSK_DINF_QCQO_REFORMULATE_TIME`
Time spent with conic quadratic reformulation.

`MSK_DINF_QCQO_REFORMULATE_WORST_CHOLESKY_COLUMN_SCALING`
Worst Cholesky column scaling.

`MSK_DINF_QCQO_REFORMULATE_WORST_CHOLESKY_DIAG_SCALING`
Worst Cholesky diagonal scaling.

`MSK_DINF_READ_DATA_TIME`
Time spent reading the data file.

`MSK_DINF_REMOTE_TIME`
The total real time in seconds spent when optimizing on a server by the process performing the optimization on the server

`MSK_DINF_SIM_DUAL_TIME`
Time spent in the dual simplex optimizer since invoking it.

`MSK_DINF_SIM_FEAS`
Feasibility measure reported by the simplex optimizer.

`MSK_DINF_SIM_OBJ`
Objective value reported by the simplex optimizer.

`MSK_DINF_SIM_PRIMAL_TIME`
Time spent in the primal simplex optimizer since invoking it.

`MSK_DINF_SIM_TIME`
Time spent in the simplex optimizer since invoking it.

`MSK_DINF_SOL_BAS_DUAL_OBJ`
Dual objective value of the basic solution. Updated if `MSK_IPAR_AUTO_UPDATE_SOL_INFO` is set .

`MSK_DINF_SOL_BAS_DVIOLCON`
Maximal dual bound violation for x^c in the basic solution. Updated if `MSK_IPAR_AUTO_UPDATE_SOL_INFO` is set .

`MSK_DINF_SOL_BAS_DVIOLVAR`
Maximal dual bound violation for x^x in the basic solution. Updated if `MSK_IPAR_AUTO_UPDATE_SOL_INFO` is set .

MSK_DINF_SOL_BAS_NRM_BARX
Infinity norm of \overline{X} in the basic solution.

MSK_DINF_SOL_BAS_NRM_SLC
Infinity norm of s_l^c in the basic solution.

MSK_DINF_SOL_BAS_NRM_SLX
Infinity norm of s_l^x in the basic solution.

MSK_DINF_SOL_BAS_NRM_SUC
Infinity norm of s_u^c in the basic solution.

MSK_DINF_SOL_BAS_NRM_SUX
Infinity norm of s_u^x in the basic solution.

MSK_DINF_SOL_BAS_NRM_XC
Infinity norm of x^c in the basic solution.

MSK_DINF_SOL_BAS_NRM_XX
Infinity norm of x^x in the basic solution.

MSK_DINF_SOL_BAS_NRM_Y
Infinity norm of y in the basic solution.

MSK_DINF_SOL_BAS_PRIMAL_OBJ
Primal objective value of the basic solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_BAS_PVIOLCON
Maximal primal bound violation for x^c in the basic solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_BAS_PVIOLVAR
Maximal primal bound violation for x^x in the basic solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITG_NRM_BARX
Infinity norm of \overline{X} in the integer solution.

MSK_DINF_SOL_ITG_NRM_XC
Infinity norm of x^c in the integer solution.

MSK_DINF_SOL_ITG_NRM_XX
Infinity norm of x^x in the integer solution.

MSK_DINF_SOL_ITG_PRIMAL_OBJ
Primal objective value of the integer solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITG_PVIOLACC
Maximal primal violation for affine conic constraints in the integer solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITG_PVIOLBARVAR
Maximal primal bound violation for \overline{X} in the integer solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITG_PVIOLCON
Maximal primal bound violation for x^c in the integer solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITG_PVIOLCONES
Maximal primal violation for primal conic constraints in the integer solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITG_PVIOLDJC
Maximal primal violation for disjunctive constraints in the integer solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITG_PVIOLITG
Maximal violation for the integer constraints in the integer solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITG_PVIOLVAR
Maximal primal bound violation for x^x in the integer solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_DUAL_OBJ
Dual objective value of the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_DVIOLACC
Maximal dual violation for the affine conic constraints in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_DVIOLBARVAR
Maximal dual bound violation for \bar{X} in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_DVIOLCON
Maximal dual bound violation for x^c in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_DVIOLCONES
Maximal dual violation for conic constraints in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_DVIOLVAR
Maximal dual bound violation for x^x in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_NRM_BARS
Infinity norm of \bar{S} in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_BARX
Infinity norm of \bar{X} in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_SLC
Infinity norm of s_l^c in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_SLX
Infinity norm of s_l^x in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_SNX
Infinity norm of s_n^x in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_SUC
Infinity norm of s_u^c in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_SUX
Infinity norm of s_u^X in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_XC
Infinity norm of x^c in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_XX
Infinity norm of x^x in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_Y
Infinity norm of y in the interior-point solution.

MSK_DINF_SOL_ITR_PRIMAL_OBJ
Primal objective value of the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_PVIOLACC
Maximal primal violation for affine conic constraints in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_PVIOLBARVAR
Maximal primal bound violation for \bar{X} in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_PVIOLCON
Maximal primal bound violation for x^c in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_PVIOLCONES
Maximal primal violation for conic constraints in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_PVIOLVAR
Maximal primal bound violation for x^x in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_TO_CONIC_TIME
Time spent in the last to conic reformulation.

MSK_DINF_WRITE_DATA_TIME
Time spent writing the data file.

11.4.22 License feature

MSK_FEATURE_PTS

Base system.

MSK_FEATURE_PTON

Conic extension.

11.4.23 Long integer information items.

MSK_LIINF_ANA_PRO_SCALARIZED_CONSTRAINT_MATRIX_NUM_COLUMNS

Number of columns in the scalarized constraint matrix.

MSK_LIINF_ANA_PRO_SCALARIZED_CONSTRAINT_MATRIX_NUM_NZ

Number of non-zero entries in the scalarized constraint matrix.

MSK_LIINF_ANA_PRO_SCALARIZED_CONSTRAINT_MATRIX_NUM_ROWS

Number of rows in the scalarized constraint matrix.

MSK_LIINF_BI_CLEAN_DUAL_DEG_ITER

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

MSK_LIINF_BI_CLEAN_DUAL_ITER

Number of dual clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_PRIMAL_DEG_ITER

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

MSK_LIINF_BI_CLEAN_PRIMAL_ITER

Number of primal clean iterations performed in the basis identification.

MSK_LIINF_BI_DUAL_ITER

Number of dual pivots performed in the basis identification.

MSK_LIINF_BI_PRIMAL_ITER

Number of primal pivots performed in the basis identification.

MSK_LIINF_INTPNT_FACTOR_NUM_NZ

Number of non-zeros in factorization.

MSK_LIINF_MIO_ANZ

Number of non-zero entries in the constraint matrix of the problem to be solved by the mixed-integer optimizer.

MSK_LIINF_MIO_INTPNT_ITER

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

MSK_LIINF_MIO_NUM_DUAL_ILLPOSED_CER

Number of dual illposed certificates encountered by the mixed-integer optimizer.

MSK_LIINF_MIO_NUM_PRIM_ILLPOSED_CER

Number of primal illposed certificates encountered by the mixed-integer optimizer.

MSK_LIINF_MIO_PRE SOLVED_ANZ

Number of non-zero entries in the constraint matrix of the problem after the mixed-integer optimizer's presolve.

MSK_LIINF_MIO_SIMPLEX_ITER

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

MSK_LIINF_RD_NUMACC

Number of affine conic constraints.

MSK_LIINF_RD_NUMANZ

Number of non-zeros in A that is read.

MSK_LIINF_RD_NUMDJC

Number of disjunctive constraints.

MSK_LIINF_RD_NUMQNZ

Number of Q non-zeros.

MSK_LIINF_SIMPLEX_ITER

Number of iterations performed by the simplex optimizer.

11.4.24 Integer information items.

MSK_IINF_ANA_PRO_NUM_CON

Number of constraints in the problem.

MSK_IINF_ANA_PRO_NUM_CON_EQ

Number of equality constraints.

MSK_IINF_ANA_PRO_NUM_CON_FR

Number of unbounded constraints.

MSK_IINF_ANA_PRO_NUM_CON_LO

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

MSK_IINF_ANA_PRO_NUM_CON_RA

Number of constraints with finite lower and upper bounds.

MSK_IINF_ANA_PRO_NUM_CON_UP

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

MSK_IINF_ANA_PRO_NUM_VAR

Number of variables in the problem.

MSK_IINF_ANA_PRO_NUM_VAR_BIN

Number of binary (0-1) variables.

MSK_IINF_ANA_PRO_NUM_VAR_CONT

Number of continuous variables.

MSK_IINF_ANA_PRO_NUM_VAR_EQ

Number of fixed variables.

MSK_IINF_ANA_PRO_NUM_VAR_FR

Number of free variables.

MSK_IINF_ANA_PRO_NUM_VAR_INT

Number of general integer variables.

MSK_IINF_ANA_PRO_NUM_VAR_LO

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

MSK_IINF_ANA_PRO_NUM_VAR_RA

Number of variables with finite lower and upper bounds.

MSK_IINF_ANA_PRO_NUM_VAR_UP

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

MSK_IINF_INTPNT_FACTOR_DIM_DENSE

Dimension of the dense sub system in factorization.

MSK_IINF_INTPNT_ITER

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

MSK_IINF_INTPNT_NUM_THREADS

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

MSK_IINF_INTPNT_SOLVE_DUAL

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

MSK_IINF_MIO_ABSGAP_SATISFIED

Non-zero if absolute gap is within tolerances.

MSK_IINF_MIO_CLIQUETABLE_SIZE

Size of the clique table.

MSK_IINF_MIO_CONSTRUCT_SOLUTION

This item informs if **MOSEK** constructed an initial integer feasible solution.

- -1: tried, but failed,
- 0: no partial solution supplied by the user,
- 1: constructed feasible solution.

MSK_IINF_MIO_INITIAL_FEASIBLE_SOLUTION

This item informs if **MOSEK** found the solution provided by the user to be feasible

- 0: solution provided by the user was not found to be feasible for the current problem,
- 1: user provided solution was found to be feasible.

MSK_IINF_MIO_NODE_DEPTH

Depth of the last node solved.

MSK_IINF_MIO_NUM_ACTIVE_NODES
 Number of active branch and bound nodes.

MSK_IINF_MIO_NUM_BRANCH
 Number of branches performed during the optimization.

MSK_IINF_MIO_NUM CLIQUE_CUTS
 Number of clique cuts.

MSK_IINF_MIO_NUM_CMIR_CUTS
 Number of Complemented Mixed Integer Rounding (CMIR) cuts.

MSK_IINF_MIO_NUM_GOMORY_CUTS
 Number of Gomory cuts.

MSK_IINF_MIO_NUM IMPLIED_BOUND_CUTS
 Number of implied bound cuts.

MSK_IINF_MIO_NUM_INT_SOLUTIONS
 Number of integer feasible solutions that have been found.

MSK_IINF_MIO_NUM_KNAPSACK_COVER_CUTS
 Number of clique cuts.

MSK_IINF_MIO_NUM_LIPRO_CUTS
 Number of lift-and-project cuts.

MSK_IINF_MIO_NUM_RELAX
 Number of relaxations solved during the optimization.

MSK_IINF_MIO_NUM_REPEATED_PRESOLVE
 Number of times presolve was repeated at root.

MSK_IINF_MIO_NUMBIN
 Number of binary variables in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMBINCONEVAR
 Number of binary cone variables in the problem to be solved by the mixed-integer optimizer.

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

MSK_IINF_MIO_NUMCONE
 Number of cones in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMCONEVAR
 Number of cone variables in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMCONT
 Number of continuous variables in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMCONTCONEVAR
 Number of continuous cone variables in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMDEXPCONES
 Number of dual exponential cones in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMDJC
 Number of disjunctive constraints in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMDPOWCONES
 Number of dual power cones in the problem to be solved by the mixed-integer optimizer.

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

MSK_IINF_MIO_NUMINTCONEVAR
 Number of integer cone variables in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMPEXPONES
 Number of primal exponential cones in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMPPOWCONES
 Number of primal power cones in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMQCONES
 Number of quadratic cones in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMRQCONES
 Number of rotated quadratic cones in the problem to be solved by the mixed-integer optimizer.

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

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

MSK_IINF_MIO_PRE SOLVED_NUMBIN
Number of binary variables in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMBIN CONEVAR
Number of binary cone variables in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMCON
Number of constraints in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMCONE
Number of cones in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMCONEVAR
Number of cone variables in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMCONT
Number of continuous variables in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMCONT CONEVAR
Number of continuous cone variables in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMDEXPCONES
Number of dual exponential cones in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMDJC
Number of disjunctive constraints in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMDPOWCONES
Number of dual power cones in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMINT
Number of integer variables in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMINT CONEVAR
Number of integer cone variables in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMPEXP CONES
Number of primal exponential cones in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMPPOWCONES
Number of primal power cones in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMQCONES
Number of quadratic cones in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMRQCONES
Number of rotated quadratic cones in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMVAR
Number of variables in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_RELGAP_SATISFIED
Non-zero if relative gap is within tolerances.

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

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

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

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

MSK_IINF_OPTIMIZE_RESPONSE
The response code returned by optimize.

MSK_IINF_PRE SOLVE_NUM_PRIMAL_PERTURBATIONS
Number perturbations to the bounds of the primal problem.

MSK_IINF_PURIFY_DUAL_SUCCESS
Is nonzero if the dual solution is purified.

MSK_IINF_PURIFY_PRIMAL_SUCCESS
Is nonzero if the primal solution is purified.

MSK_IINF_RD_NUMBARVAR
Number of symmetric variables read.

MSK_IINF_RD_NUMCON
Number of constraints read.

MSK_IINF_RD_NUMCONE
Number of conic constraints read.

MSK_IINF_RD_NUMINTVAR
 Number of integer-constrained variables read.

MSK_IINF_RD_NUMQ
 Number of nonempty Q matrices read.

MSK_IINF_RD_NUMVAR
 Number of variables read.

MSK_IINF_RD_PROTOTYPE
 Problem type.

MSK_IINF_SIM_DUAL_DEG_ITER
 The number of dual degenerate iterations.

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

MSK_IINF_SIM_DUAL_HOTSTART_LU
 If 1 then a valid basis factorization of full rank was located and used by the dual simplex algorithm.

MSK_IINF_SIM_DUAL_INF_ITER
 The number of iterations taken with dual infeasibility.

MSK_IINF_SIM_DUAL_ITER
 Number of dual simplex iterations during the last optimization.

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

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

MSK_IINF_SIM_PRIMAL_DEG_ITER
 The number of primal degenerate iterations.

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

MSK_IINF_SIM_PRIMAL_HOTSTART_LU
 If 1 then a valid basis factorization of full rank was located and used by the primal simplex algorithm.

MSK_IINF_SIM_PRIMAL_INF_ITER
 The number of iterations taken with primal infeasibility.

MSK_IINF_SIM_PRIMAL_ITER
 Number of primal simplex iterations during the last optimization.

MSK_IINF_SIM_SOLVE_DUAL
 Is non-zero if dual problem is solved.

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

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

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

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

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

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

MSK_IINF_STO_NUM_A_REALLOC
 Number of times the storage for storing A has been changed. A large value may indicate that memory fragmentation may occur.

11.4.25 Information item types

MSK_INF_DOU_TYPE

Is a double information type.

MSK_INF_INT_TYPE

Is an integer.

MSK_INF_LINT_TYPE

Is a long integer.

11.4.26 Input/output modes

MSK_IOMODE_READ

The file is read-only.

MSK_IOMODE_WRITE

The file is write-only. If the file exists then it is truncated when it is opened. Otherwise it is created when it is opened.

MSK_IOMODE_READWRITE

The file is to read and write.

11.4.27 Specifies the branching direction.

MSK_BRANCH_DIR_FREE

The mixed-integer optimizer decides which branch to choose.

MSK_BRANCH_DIR_UP

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

MSK_BRANCH_DIR_DOWN

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

MSK_BRANCH_DIR_NEAR

Branch in direction nearest to selected fractional variable.

MSK_BRANCH_DIR_FAR

Branch in direction farthest from selected fractional variable.

MSK_BRANCH_DIR_ROOT_LP

Chose direction based on root lp value of selected variable.

MSK_BRANCH_DIR_GUIDED

Branch in direction of current incumbent.

MSK_BRANCH_DIR_PSEUDOCOST

Branch based on the pseudocost of the variable.

11.4.28 Specifies the reformulation method for mixed-integer quadratic problems.

MSK_MIO_QCQO_REFORMULATION_METHOD_FREE

The mixed-integer optimizer decides which reformulation method to apply.

MSK_MIO_QCQO_REFORMULATION_METHOD_NONE

No reformulation method is applied.

MSK_MIO_QCQO_REFORMULATION_METHOD_LINEARIZATION

A reformulation via linearization is applied.

MSK_MIO_QCQO_REFORMULATION_METHOD_EIGEN_VAL_METHOD

The eigenvalue method is applied.

MSK_MIO_QCQO_REFORMULATION_METHOD_DIAG_SDP

A perturbation of matrix diagonals via the solution of SDPs is applied.

MSK_MIO_QCQO_REFORMULATION_METHOD_RELAX_SDP

A Reformulation based on the solution of an SDP-relaxation of the problem is applied.

11.4.29 Specifies the problem data permutation method for mixed-integer problems.

MSK_MIO_DATA_PERMUTATION_METHOD_NONE

No problem data permutation is applied.

MSK_MIO_DATA_PERMUTATION_METHOD_CYCLIC_SHIFT

A random cyclic shift is applied to permute the problem data.

MSK_MIO_DATA_PERMUTATION_METHOD_RANDOM

A random permutation is applied to the problem data.

11.4.30 Continuous mixed-integer solution type

MSK_MIO_CONT_SOL_NONE

No interior-point or basic solution are reported when the mixed-integer optimizer is used.

MSK_MIO_CONT_SOL_ROOT

The reported interior-point and basic solutions are a solution to the root node problem when mixed-integer optimizer is used.

MSK_MIO_CONT_SOL_ITG

The reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. A solution is only reported in case the problem has a primal feasible solution.

MSK_MIO_CONT_SOL_ITG_REL

In case the problem is primal feasible then the reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. If the problem is primal infeasible, then the solution to the root node problem is reported.

11.4.31 Integer restrictions

MSK_MIO_MODE_IGNORED

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

MSK_MIO_MODE_SATISFIED

Integer restrictions should be satisfied.

11.4.32 Mixed-integer node selection types

MSK_MIO_NODE_SELECTION_FREE

The optimizer decides the node selection strategy.

MSK_MIO_NODE_SELECTION_FIRST

The optimizer employs a depth first node selection strategy.

MSK_MIO_NODE_SELECTION_BEST

The optimizer employs a best bound node selection strategy.

MSK_MIO_NODE_SELECTION_PSEUDO

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

11.4.33 MPS file format type

MSK_MPS_FORMAT_STRICT

It is assumed that the input file satisfies the MPS format strictly.

MSK_MPS_FORMAT_RELAXED

It is assumed that the input file satisfies a slightly relaxed version of the MPS format.

MSK_MPS_FORMAT_FREE

It is assumed that the input file satisfies the free MPS format. This implies that spaces are not allowed in names. Otherwise the format is free.

MSK_MPS_FORMAT_CPLEX

The CPLEX compatible version of the MPS format is employed.

11.4.34 Objective sense types

MSK_OBJECTIVE_SENSE_MINIMIZE

The problem should be minimized.

MSK_OBJECTIVE_SENSE_MAXIMIZE

The problem should be maximized.

11.4.35 On/off

MSK_ON

Switch the option on.

MSK_OFF

Switch the option off.

11.4.36 Optimizer types

MSK_OPTIMIZER_CONIC

The optimizer for problems having conic constraints.

MSK_OPTIMIZER_DUAL_SIMPLEX

The dual simplex optimizer is used.

MSK_OPTIMIZER_FREE

The optimizer is chosen automatically.

MSK_OPTIMIZER_FREE_SIMPLEX

One of the simplex optimizers is used.

MSK_OPTIMIZER_INTPNT

The interior-point optimizer is used.

MSK_OPTIMIZER_MIXED_INT

The mixed-integer optimizer.

MSK_OPTIMIZER_PRIMAL_SIMPLEX

The primal simplex optimizer is used.

11.4.37 Ordering strategies

MSK_ORDER_METHOD_FREE

The ordering method is chosen automatically.

MSK_ORDER_METHOD_APPMINLOC

Approximate minimum local fill-in ordering is employed.

MSK_ORDER_METHOD_EXPERIMENTAL

This option should not be used.

MSK_ORDER_METHOD_TRY_GRAPHPAR

Always try the graph partitioning based ordering.

MSK_ORDER_METHOD_FORCE_GRAPHPAR

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

MSK_ORDER_METHOD_NONE

No ordering is used.

11.4.38 Presolve method.

MSK_PRESOLVE_MODE_OFF

The problem is not presolved before it is optimized.

MSK_PRESOLVE_MODE_ON

The problem is presolved before it is optimized.

MSK_PRESOLVE_MODE_FREE

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

11.4.39 Parameter type

MSK_PAR_INVALID_TYPE
Not a valid parameter.

MSK_PAR_DOU_TYPE
Is a double parameter.

MSK_PAR_INT_TYPE
Is an integer parameter.

MSK_PAR_STR_TYPE
Is a string parameter.

11.4.40 Problem data items

MSK_PI_VAR
Item is a variable.

MSK_PI_CON
Item is a constraint.

MSK_PI_CONE
Item is a cone.

11.4.41 Problem types

MSK_PROBTYPE_LO
The problem is a linear optimization problem.

MSK_PROBTYPE_QO
The problem is a quadratic optimization problem.

MSK_PROBTYPE_QCQO
The problem is a quadratically constrained optimization problem.

MSK_PROBTYPE_CONIC
A conic optimization.

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

11.4.42 Problem status keys

MSK_PRO_STA_UNKNOWN
Unknown problem status.

MSK_PRO_STA_PRIM_AND_DUAL_FEAS
The problem is primal and dual feasible.

MSK_PRO_STA_PRIM_FEAS
The problem is primal feasible.

MSK_PRO_STA_DUAL_FEAS
The problem is dual feasible.

MSK_PRO_STA_PRIM_INFEAS
The problem is primal infeasible.

MSK_PRO_STA_DUAL_INFEAS
The problem is dual infeasible.

MSK_PRO_STA_PRIM_AND_DUAL_INFEAS
The problem is primal and dual infeasible.

MSK_PRO_STA_ILL_POSED
The problem is ill-posed. For example, it may be primal and dual feasible but have a positive duality gap.

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

11.4.43 XML writer output mode

MSK_WRITE_XML_MODE_ROW

Write in row order.

MSK_WRITE_XML_MODE_COL

Write in column order.

11.4.44 Response code type

MSK_RESPONSE_OK

The response code is OK.

MSK_RESPONSE_WRN

The response code is a warning.

MSK_RESPONSE_TRM

The response code is an optimizer termination status.

MSK_RESPONSE_ERR

The response code is an error.

MSK_RESPONSE_UNK

The response code does not belong to any class.

11.4.45 Scaling type

MSK_SCALING_FREE

The optimizer chooses the scaling heuristic.

MSK_SCALING_NONE

No scaling is performed.

11.4.46 Scaling method

MSK_SCALING_METHOD_POW2

Scales only with power of 2 leaving the mantissa untouched.

MSK_SCALING_METHOD_FREE

The optimizer chooses the scaling heuristic.

11.4.47 Sensitivity types

MSK_SENSITIVITY_TYPE_BASIS

Basis sensitivity analysis is performed.

11.4.48 Simplex selection strategy

MSK_SIM_SELECTION_FREE

The optimizer chooses the pricing strategy.

MSK_SIM_SELECTION_FULL

The optimizer uses full pricing.

MSK_SIM_SELECTION_ASE

The optimizer uses approximate steepest-edge pricing.

MSK_SIM_SELECTION_DEVEX

The optimizer uses devex steepest-edge pricing (or if it is not available an approximate steep-edge selection).

MSK_SIM_SELECTION_SE

The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

MSK_SIM_SELECTION_PARTIAL

The optimizer uses a partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.

11.4.49 Solution items

MSK_SOL_ITEM_XC

Solution for the constraints.

MSK_SOL_ITEM_XX

Variable solution.

MSK_SOL_ITEM_Y

Lagrange multipliers for equations.

MSK_SOL_ITEM_SLC

Lagrange multipliers for lower bounds on the constraints.

MSK_SOL_ITEM_SUC

Lagrange multipliers for upper bounds on the constraints.

MSK_SOL_ITEM_SLX

Lagrange multipliers for lower bounds on the variables.

MSK_SOL_ITEM_SUX

Lagrange multipliers for upper bounds on the variables.

MSK_SOL_ITEM_SNX

Lagrange multipliers corresponding to the conic constraints on the variables.

11.4.50 Solution status keys

MSK_SOL_STA_UNKNOWN

Status of the solution is unknown.

MSK_SOL_STA_OPTIMAL

The solution is optimal.

MSK_SOL_STA_PRIM_FEAS

The solution is primal feasible.

MSK_SOL_STA_DUAL_FEAS

The solution is dual feasible.

MSK_SOL_STA_PRIM_AND_DUAL_FEAS

The solution is both primal and dual feasible.

MSK_SOL_STA_PRIM_INFEAS_CER

The solution is a certificate of primal infeasibility.

MSK_SOL_STA_DUAL_INFEAS_CER

The solution is a certificate of dual infeasibility.

MSK_SOL_STA_PRIM_ILLPOSED_CER

The solution is a certificate that the primal problem is illposed.

MSK_SOL_STA_DUAL_ILLPOSED_CER

The solution is a certificate that the dual problem is illposed.

MSK_SOL_STA_INTEGER_OPTIMAL

The primal solution is integer optimal.

11.4.51 Solution types

MSK_SOL_BAS

The basic solution.

MSK_SOL_ITR

The interior solution.

MSK_SOL_ITG

The integer solution.

11.4.52 Solve primal or dual form

MSK_SOLVE_FREE

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

MSK_SOLVE_PRIMAL

The optimizer should solve the primal problem.

MSK_SOLVE_DUAL

The optimizer should solve the dual problem.

11.4.53 Status keys

MSK_SK_UNK

The status for the constraint or variable is unknown.

MSK_SK_BAS

The constraint or variable is in the basis.

MSK_SK_SUPBAS

The constraint or variable is super basic.

MSK_SK_LOW

The constraint or variable is at its lower bound.

MSK_SK_UPR

The constraint or variable is at its upper bound.

MSK_SK_FIX

The constraint or variable is fixed.

MSK_SK_INF

The constraint or variable is infeasible in the bounds.

11.4.54 Starting point types

MSK_STARTING_POINT_FREE

The starting point is chosen automatically.

MSK_STARTING_POINT_GUESS

The optimizer guesses a starting point.

MSK_STARTING_POINT_CONSTANT

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

MSK_STARTING_POINT_SATISFY_BOUNDS

The starting point is 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.

11.4.55 Stream types

MSK_STREAM_LOG

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

MSK_STREAM_MSG

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

MSK_STREAM_ERR

Error stream. Error messages are written to this stream.

MSK_STREAM_WRN

Warning stream. Warning messages are written to this stream.

11.4.56 Integer values

MSK_MAX_STR_LEN

Maximum string length allowed in **MOSEK**.

MSK_LICENSE_BUFFER_LENGTH

The length of a license key buffer.

11.4.57 Variable types

MSK_VAR_TYPE_CONT

Is a continuous variable.

MSK_VAR_TYPE_INT

Is an integer variable.

11.5 Supported domains

This section lists the domains supported by **MOSEK**.

11.5.1 Linear domains

Each linear domain is determined by the dimension n .

- : the **zero domain**, consisting of the origin $0^n \in \mathbb{R}^n$.
- : the **nonnegative orthant domain** $\mathbb{R}_{\geq 0}^n$.
- : the **nonpositive orthant domain** $\mathbb{R}_{\leq 0}^n$.
- : the **free domain**, consisting of the whole \mathbb{R}^n .

Membership in a linear domain is equivalent to imposing the corresponding set of n linear constraints, for instance $Fx + g \in 0^n$ is equivalent to $Fx + g = 0$ and so on. The free domain imposes no restriction.

11.5.2 Quadratic cone domains

The quadratic domains are determined by the dimension n .

- : the **quadratic cone domain** is the subset of \mathbb{R}^n defined as

$$\mathcal{Q}^n = \left\{ x \in \mathbb{R}^n : x_1 \geq \sqrt{x_2^2 + \cdots + x_n^2} \right\}.$$

- : the **rotated quadratic cone domain** is the subset of \mathbb{R}^n defined as

$$\mathcal{Q}_r^n = \left\{ x \in \mathbb{R}^n : 2x_1x_2 \geq x_3^2 + \cdots + x_n^2, x_1, x_2 \geq 0 \right\}.$$

11.5.3 Exponential cone domains

- : the **primal exponential cone domain** is the subset of \mathbb{R}^3 defined as

$$K_{\text{exp}} = \left\{ (x_1, x_2, x_3) \in \mathbb{R}^3 : x_1 \geq x_2 \exp(x_3/x_2), x_1, x_2 \geq 0 \right\}.$$

- : the **dual exponential cone domain** is the subset of \mathbb{R}^3 defined as

$$K_{\text{exp}}^* = \left\{ (x_1, x_2, x_3) \in \mathbb{R}^3 : x_1 \leq -x_3 \exp(x_2/x_3 - 1), x_1 \geq 0, x_3 \leq 0 \right\}.$$

11.5.4 Power cone domains

A power cone domain is determined by the dimension n and a sequence of $1 \leq n_l < n$ positive real numbers (weights) $\alpha_1, \dots, \alpha_{n_l}$.

- : the **primal power cone domain** is the subset of \mathbb{R}^n defined as

$$\mathcal{P}_n^{(\alpha_1, \dots, \alpha_{n_l})} = \left\{ x \in \mathbb{R}^n : \prod_{i=1}^{n_l} x_i^{\beta_i} \geq \sqrt{x_{n_l+1}^2 + \dots + x_n^2}, x_1, \dots, x_{n_l} \geq 0 \right\}.$$

where β_i are the weights normalized to add up to 1, ie. $\beta_i = \alpha_i / (\sum_j \alpha_j)$ for $i = 1, \dots, n_l$. The name n_l reads as “n left”, the length of the product on the left-hand side of the definition.

- : the **dual power cone domain** is the subset of \mathbb{R}^n defined as

$$\left(\mathcal{P}_n^{(\alpha_1, \dots, \alpha_{n_l})} \right)^* = \left\{ x \in \mathbb{R}^n : \prod_{i=1}^{n_l} \left(\frac{x_i}{\beta_i} \right)^{\beta_i} \geq \sqrt{x_{n_l+1}^2 + \dots + x_n^2}, x_1, \dots, x_{n_l} \geq 0 \right\}.$$

where β_i are the weights normalized to add up to 1, ie. $\beta_i = \alpha_i / (\sum_j \alpha_j)$ for $i = 1, \dots, n_l$. The name n_l reads as “n left”, the length of the product on the left-hand side of the definition.

- **Remark:** in MOSEK 9 power cones were available only in the special case with $n_l = 2$ and weights $(\alpha, 1 - \alpha)$ for some $0 < \alpha < 1$ specified as cone parameter.

11.5.5 Geometric mean cone domains

A geometric mean cone domain is determined by the dimension n .

- : the **primal geometric mean cone domain** is the subset of \mathbb{R}^n defined as

$$\mathcal{GM}^n = \left\{ x \in \mathbb{R}^n : \left(\prod_{i=1}^{n-1} x_i \right)^{1/(n-1)} \geq |x_n|, x_1, \dots, x_{n-1} \geq 0 \right\}.$$

It is a special case of the primal power cone domain with $n_l = n - 1$ and weights $\alpha = (1, \dots, 1)$.

- : the **dual geometric mean cone domain** is the subset of \mathbb{R}^n defined as

$$(\mathcal{GM}^n)^* = \left\{ x \in \mathbb{R}^n : (n-1) \left(\prod_{i=1}^{n-1} x_i \right)^{1/(n-1)} \geq |x_n|, x_1, \dots, x_{n-1} \geq 0 \right\}.$$

It is a special case of the dual power cone domain with $n_l = n - 1$ and weights $\alpha = (1, \dots, 1)$.

11.5.6 Vectorized semidefinite domain

- : the **vectorized PSD cone domain** is determined by the dimension n , which must be of the form $n = d(d+1)/2$. Then the domain is defined as

$$\mathcal{S}_+^{d, \text{vec}} = \{ (x_1, \dots, x_{d(d+1)/2}) \in \mathbb{R}^n : \text{sMat}(x) \in \mathcal{S}_+^d \},$$

where

$$\text{sMat}(x) = \begin{bmatrix} x_1 & x_2/\sqrt{2} & \cdots & x_d/\sqrt{2} \\ x_2/\sqrt{2} & x_{d+1} & \cdots & x_{2d-1}/\sqrt{2} \\ \cdots & \cdots & \cdots & \cdots \\ x_d/\sqrt{2} & x_{2d-1}/\sqrt{2} & \cdots & x_{d(d+1)/2} \end{bmatrix},$$

or equivalently

$$\mathcal{S}_+^{d,\text{vec}} = \{\text{sVec}(X) : X \in \mathcal{S}_+^d\},$$

where

$$\text{sVec}(X) = (X_{11}, \sqrt{2}X_{21}, \dots, \sqrt{2}X_{d1}, X_{22}, \sqrt{2}X_{32}, \dots, X_{dd}).$$

In other words, the domain consists of vectorizations of the lower-triangular part of a positive semidefinite matrix, with the non-diagonal elements additionally rescaled.

Chapter 12

Supported File Formats

MOSEK supports a range of problem and solution formats listed in [Table 12.1](#) and [Table 12.2](#).

The most important are:

- the **Task format**, **MOSEK**'s native binary format which supports all features that **MOSEK** supports. It is the closest possible representation of the internal data in a task and it is ideal for submitting problem data support questions.
- the **PTF format**, **MOSEK**'s human-readable format that supports all linear, conic and mixed-integer features. It is ideal for debugging. It is not an exact copy of all the data in the task, but it contains all information required to reconstruct it, presented in a readable fashion.
- **MPS**, **LP**, **CBF** formats are industry standards, each supporting some limited set of features, and potentially requiring some degree of reformulation during read/write.

Problem formats

Table 12.1: List of supported file formats for optimization problems.

Format Type	Ext.	Binary/Text	LP	QCQO	ACC	SDP	DJC	Sol	Param
<i>LP</i>	lp	plain text	X	X					
<i>MPS</i>	mps	plain text	X	X					
<i>PTF</i>	ptf	plain text	X		X	X	X	X	
<i>CBF</i>	cbf	plain text	X		X	X			
<i>Task format</i>	task	binary	X	X	X	X	X	X	X
<i>Jtask format</i>	jtask	text/JSON	X	X	X	X	X	X	X
<i>OPF</i> (deprecated for conic problems)	opf	plain text	X	X				X	X

The columns of the table indicate if the specified file format supports:

- LP - linear problems,
- QCQO - quadratic objective or constraints,
- ACC - affine conic constraints,
- SDP - semidefinite cone/variables,
- DJC - disjunctive constraints,
- Sol - solutions,
- Param - optimizer parameters.

Solution formats

Table 12.2: List of supported solution formats.

Format Type	Ext.	Binary/Text	Description
<i>SOL</i>	sol	plain text	Interior Solution
	bas	plain text	Basic Solution
	int	plain text	Integer
<i>Jsol format</i>	jsol	text/JSON	All solutions

Compression

MOSEK supports GZIP and Zstandard compression. Problem files with extension `.gz` (for GZIP) and `.zst` (for Zstandard) are assumed to be compressed when read, and are automatically compressed when written. For example, a file called

`problem.mps.zst`

will be considered as a Zstandard compressed MPS file.

12.1 The LP File Format

MOSEK supports the LP file format with some extensions. The LP format is not a completely well-defined standard and hence different optimization packages may interpret the same LP file in slightly different ways. **MOSEK** tries to emulate as closely as possible CPLEX's behavior, but tries to stay backward compatible.

The LP file format can specify problems of the form

$$\begin{aligned}
 & \text{minimize/maximize} && c^T x + \frac{1}{2} q^o(x) \\
 & \text{subject to} && l^c \leq Ax + \frac{1}{2} q(x) \leq u^c, \\
 & && l^x \leq x \leq u^x, \\
 & && x_{\mathcal{J}} \text{ integer},
 \end{aligned}$$

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

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

- $A \in \mathbb{R}^{m \times n}$ 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

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

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

12.1.1 File 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.

Objective Function

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.

$$c^T x + \frac{1}{2} x^T Q x.$$

The objective may be given a name by writing

```
myname:
```

before the expressions. If no name is given, then the objective is named **obj**.

The objective function contains linear and quadratic terms. The linear terms are written as

```
4 x1 + x2 - 0.1 x3
```

and so forth. The quadratic terms are written in square brackets (`[]/2`) 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.

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 per line, but **MOSEK** supports defining ranged constraints by using double-colon ($::$) instead of a single-colon ($:$) after the constraint name, i.e.

$$-5 \leq x_1 + x_2 \leq 5 \quad (12.1)$$

may be written as

```
con:: -5 < x_1 + x_2 < 5
```

By default **MOSEK** writes ranged constraints this way.

If the files must adhere to the LP standard, ranged constraints must either be split into upper bounded and lower bounded constraints or be written as an equality with a slack variable. For example the expression (12.1) may be written as

$$x_1 + x_2 - sl_1 = 0, \quad -5 \leq sl_1 \leq 5.$$

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
```

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.

Terminating Section

Finally, an LP formatted file must be terminated with the keyword

```
end
```

12.1.2 LP File Examples

Linear example lo1.lp

```
\ File: lo1.lp
maximize
obj: 3 x1 + x2 + 5 x3 + x4
subject to
c1: 3 x1 + x2 + 2 x3 = 30
c2: 2 x1 + x2 + 3 x3 + x4 >= 15
c3: 2 x2 + 3 x4 <= 25
bounds
0 <= x1 <= +infinity
0 <= x2 <= 10
0 <= x3 <= +infinity
0 <= x4 <= +infinity
end
```

Mixed integer example milo1.lp

```
maximize
obj: x1 + 6.4e-01 x2
subject to
c1: 5e+01 x1 + 3.1e+01 x2 <= 2.5e+02
c2: 3e+00 x1 - 2e+00 x2 >= -4e+00
bounds
0 <= x1 <= +infinity
0 <= x2 <= +infinity
general
x1 x2
end
```

12.1.3 LP Format peculiarities

Comments

Anything on a line after a \ is ignored and is treated as a comment.

Names

A name for an objective, a constraint or a variable may contain the letters a-z, A-Z, the digits 0-9 and the characters

!"#\$%&() / , . ; ? @ _ ' ` | ~

The first character in a name must not be a number, a period or the letter e or E. Keywords must not be used as names.

MOSEK accepts any character as valid for names, except \0. A name that is not allowed in LP file will be changed and a warning will be issued.

The algorithm for making names LP valid works as follows: The name is interpreted as an **utf-8** string. For a Unicode character *c*:

- If *c* = _ (underscore), the output is __ (two underscores).
- If *c* is a valid LP name character, the output is just *c*.
- If *c* is another character in the ASCII range, the output is _XX, where XX is the hexadecimal code for the character.
- If *c* is a character in the range 127-65535, the output is _uXXXX, where XXXX is the hexadecimal code for the character.
- If *c* is a character above 65535, the output is _UXXXXXXXX, where XXXXXXXX is the hexadecimal code for the character.

Invalid **utf-8** substrings are escaped as _XX', and if a name starts with a period, e or E, that character is escaped as _XX.

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.

12.2 The MPS File Format

MOSEK supports the standard MPS format with some extensions. For a detailed description of the MPS format see the book by Nazareth [Naz87].

12.2.1 MPS File Structure

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

$$\begin{aligned}
 &\text{maximize/minimize} && c^T x + q_0(x) \\
 &l^c \leq && Ax + q(x) \leq u^c, \\
 &l^x \leq && x \leq u^x, \\
 &&& x \in \mathcal{K}, \\
 &&& x_{\mathcal{J}} \text{ integer},
 \end{aligned} \tag{12.2}$$

where

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

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

$$q_i(x) = \frac{1}{2}x^T Q^i x$$

where it is assumed that $Q^i = (Q^i)^T$. Please note the explicit $\frac{1}{2}$ in the quadratic term and that Q^i is required to be symmetric. The same applies to q_0 .

- \mathcal{K} is a convex cone.
- $\mathcal{J} \subseteq \{1, 2, \dots, n\}$ is an index set of the integer-constrained variables.
- c is the vector of objective coefficients.

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

```
*          1          2          3          4          5          6
*23456789012345678901234567890123456789012345678901234567890
NAME          [name]
OBJSENSE
    [objsense]
OBJNAME          [objname]
ROWS
?  [cname1]
COLUMNS
    [vname1]  [cname1]  [value1]          [cname2]  [value2]
RHS
    [name]    [cname1]  [value1]          [cname2]  [value2]
RANGES
    [name]    [cname1]  [value1]          [cname2]  [value2]
QSECTION          [cname1]
    [vname1]  [vname2]  [value1]          [vname3]  [value2]
QMATRIX
    [vname1]  [vname2]  [value1]
QUADOBJ
    [vname1]  [vname2]  [value1]
QCMATRIX          [cname1]
    [vname1]  [vname2]  [value1]
BOUNDS
?? [name]    [vname1]  [value1]
CSECTION          [kname1]  [value1]          [ktype]
    [vname1]
ENDATA
```

Here the names in capitals are keywords of the MPS format and names in brackets are custom defined names or values. A couple of notes on the structure:

- Fields: All items surrounded by brackets appear in *fields*. The fields named “**valueN**” are numerical values. Hence, they must have the format

$[+|-]XXXXXXXX.XXXXXX[[e|E][+|-]XXX]$

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 specific **MOSEK** extensions of the MPS format. The sections **QMATRIX**, **QUADOBJ** and **QCMATRIX** are included for sake of compatibility with other vendors extensions to the MPS format.
- The standard MPS format is a fixed format, i.e. everything in the MPS file must be within certain fixed positions. **MOSEK** also supports a *free format*. See [Sec. 12.2.5](#) for details.

Linear example lo1.mps

A concrete example of a MPS file is presented below:

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

Subsequently each individual section in the MPS format is discussed.

NAME (optional)

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

OBJSENSE (optional)

This is an optional section that can be used to specify the sense of the objective function. The **OBJSENSE** section contains one line at most which can be one of the following:

```
MIN
MINIMIZE
MAX
MAXIMIZE
```

It should be obvious what the implication is of each of these four lines.

OBJNAME (optional)

This is an optional section that can be used to specify the name of the row that is used as objective function. **objname** should be a valid row name.

ROWS

A record in the **ROWS** section has the form

```
? [cname1]
```

where the requirements for the fields are as follows:

Field	Starting Position	Max Width	required	Description
?	2	1	Yes	Constraint key
[cname1]	5	8	Yes	Constraint name

Hence, in this section each constraint is assigned a unique name denoted by [cname1]. Please note that [cname1] starts in position 5 and the field can be at most 8 characters wide. An initial key ? must be present to specify the type of the constraint. The key can have values E, G, L, or N with the following interpretation:

Constraint type	l_i^c	u_i^c
E (equal)	finite	$= l_i^c$
G (greater)	finite	∞
L (lower)	$-\infty$	finite
N (none)	$-\infty$	∞

In the MPS format the objective vector is not specified explicitly, but one of the constraints having the key N will be used as the objective vector c . In general, if multiple N type constraints are specified, then the first will be used as the objective vector c , unless something else was specified in the section **OBJNAME**.

COLUMNS

In this section the elements of A are specified using one or more records having the form:

```
[vname1] [cname1] [value1] [cname2] [value2]
```

where the requirements for each field are as follows:

Field	Starting Position	Max Width	required	Description
[vname1]	5	8	Yes	Variable name
[cname1]	15	8	Yes	Constraint name
[value1]	25	12	Yes	Numerical value
[cname2]	40	8	No	Constraint name
[value2]	50	12	No	Numerical value

Hence, a record specifies one or two elements a_{ij} of A using the principle that [vname1] and [cname1] determines j and i respectively. Please note that [cname1] must be a constraint name specified in the

ROWS section. Finally, [value1] denotes the numerical value of a_{ij} . Another optional element is specified by [cname2], and [value2] for the variable specified by [vname1]. Some important comments are:

- All elements belonging to one variable must be grouped together.
- Zero elements of A should not be specified.
- At least one element for each variable should be specified.

RHS (optional)

A record in this section has the format

[name]	[cname1]	[value1]	[cname2]	[value2]
--------	----------	----------	----------	----------

where the requirements for each field are as follows:

Field	Starting Position	Max Width	required	Description
[name]	5	8	Yes	Name of the RHS vector
[cname1]	15	8	Yes	Constraint name
[value1]	25	12	Yes	Numerical value
[cname2]	40	8	No	Constraint name
[value2]	50	12	No	Numerical value

The interpretation of a record is that [name] is the name of the RHS vector to be specified. In general, several vectors can be specified. [cname1] denotes a constraint name previously specified in the ROWS section. Now, assume that this name has been assigned to the i -th constraint and v_1 denotes the value specified by [value1], then the interpretation of v_1 is:

Constraint	l_i^c	u_i^c
E	v_1	v_1
G	v_1	
L		v_1
N		

An optional second element is specified by [cname2] and [value2] and is interpreted in the same way. Please note that it is not necessary to specify zero elements, because elements are assumed to be zero.

RANGES (optional)

A record in this section has the form

[name]	[cname1]	[value1]	[cname2]	[value2]
--------	----------	----------	----------	----------

where the requirements for each fields are as follows:

Field	Starting Position	Max Width	required	Description
[name]	5	8	Yes	Name of the RANGE vector
[cname1]	15	8	Yes	Constraint name
[value1]	25	12	Yes	Numerical value
[cname2]	40	8	No	Constraint name
[value2]	50	12	No	Numerical value

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

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

Another constraint bound can optionally be modified using [cname2] and [value2] the same way.

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 terms belong. A record in the QSECTION has the form

[vname1]	[vname2]	[value1]	[vname3]	[value2]
----------	----------	----------	----------	----------

where the requirements for each field are:

Field	Starting Position	Max Width	required	Description
[vname1]	5	8	Yes	Variable name
[vname2]	15	8	Yes	Variable name
[value1]	25	12	Yes	Numerical value
[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

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

has the following MPS file representation

```

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

```

Regarding the 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 .

QMATRIX/QUADOBJ (optional)

The QMATRIX and QUADOBJ sections allow to define the quadratic term of the objective function. They differ in how the quadratic term of the objective function is stored:

- QMATRIX stores all the nonzeros coefficients, without taking advantage of the symmetry of the Q matrix.
- QUADOBJ stores the upper diagonal nonzero elements of the Q matrix.

A record in both sections has the form:

[vname1]	[vname2]	[value1]
----------	----------	----------

where the requirements for each field are:

Field	Starting Position	Max Width	required	Description
[vname1]	5	8	Yes	Variable name
[vname2]	15	8	Yes	Variable name
[value1]	25	12	Yes	Numerical value

A record specifies one elements of the Q matrix in the objective function. Hence, if the names [vname1] and [vname2] have been assigned to the k -th and j -th variable, then Q_{kj} is assigned the value given by [value1]. Note that a line must appear for each off-diagonal coefficient if using a QMATRIX section, while only one entry is required in a QUADOBJ section. The quadratic part of the objective function will be evaluated as $1/2x^T Qx$.

The example

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

has the following MPS file representation using QMATRIX

```

* File: qo1_matrix.mps
NAME          qo1_qmatrix
ROWS
  N  obj
  G  c1
COLUMNS
  x1      c1      1.0
  x2      obj     -1.0
  x2      c1      1.0
  x3      c1      1.0
RHS
  rhs     c1      1.0
QMATRIX
  x1      x1      2.0
  x1      x3     -1.0
  x3      x1     -1.0
  x2      x2      0.2
  x3      x3      2.0
ENDATA

```

or the following using QUADOBJ

```

* File: qo1_quadobj.mps
NAME          qo1_quadobj
ROWS
  N  obj
  G  c1
COLUMNS
  x1      c1      1.0
  x2      obj     -1.0
  x2      c1      1.0
  x3      c1      1.0
RHS
  rhs     c1      1.0
QUADOBJ
  x1      x1      2.0
  x1      x3     -1.0
  x2      x2      0.2
  x3      x3      2.0
ENDATA

```

Please also note that:

- A QMATRIX/QUADOBJ section can appear in an arbitrary order after the COLUMNS section.
- All variable names occurring in the QMATRIX/QUADOBJ section must already be specified in the COLUMNS section.

QMATRIX (optional)

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

[vname1]	[vname2]	[value1]
----------	----------	----------

where the requirements for each field are:

Field	Starting Position	Max Width	required	Description
[vname1]	5	8	Yes	Variable name
[vname2]	15	8	Yes	Variable name
[value1]	25	12	Yes	Numerical value

A record specifies an entry of the Q^i matrix where [cname1] specifies the i . Hence, if the names [vname1] and [vname2] have been assigned to the k -th and j -th variable, then Q_{kj}^i is assigned the value given by [value1]. Moreover, the quadratic term is represented as $1/2x^T Qx$.

The example

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

has the following MPS file representation

```

* File: qo1.mps
NAME          qo1
ROWS
  N  obj
  G  c1
  L  q1
COLUMNS

```

(continues on next page)

(continued from previous page)

x1	c1	1.0
x2	obj	-1.0
x2	c1	1.0
x3	c1	1.0
RHS		
rhs	c1	1.0
rhs	q1	10.0
QCMATRIX		
q1	q1	
x1	x1	2.0
x1	x3	-1.0
x3	x1	-1.0
x2	x2	0.2
x3	x3	2.0
ENDATA		

Regarding the QCMATRIXs please note that:

- Only one QCMATRIX is allowed for each constraint.
- The QCMATRIXs can appear in an arbitrary order after the COLUMNS section.
- All variable names occurring in the QSECTION must already be specified in the COLUMNS section.
- QCMATRIX does not exploit the symmetry of Q : an off-diagonal entry (i, j) should appear twice.

BOUNDS (optional)

In the BOUNDS section changes to the default bounds vectors l^x and u^x are specified. The default bounds vectors are $l^x = 0$ and $u^x = \infty$. Moreover, it is possible to specify several sets of bound vectors. A record in this section has the form

??	[name]	[vname1]	[value1]
----	--------	----------	----------

where the requirements for each field are:

Field	Starting Position	Max Width	Required	Description
??	2	2	Yes	Bound key
[name]	5	8	Yes	Name of the BOUNDS vector
[vname1]	15	8	Yes	Variable name
[value1]	25	12	No	Numerical value

Hence, a record in the BOUNDS section has the following interpretation: [name] is the name of the bound vector and [vname1] is the name of the variable for which the bounds are modified by the record. ?? and [value1] are used to modify the bound vectors according to the following table:

??	l_j^x	u_j^x	Made integer (added to \mathcal{J})
FR	$-\infty$	∞	No
FX	v_1	v_1	No
LO	v_1	unchanged	No
MI	$-\infty$	unchanged	No
PL	unchanged	∞	No
UP	unchanged	v_1	No
BV	0	1	Yes
LI	$\lceil v_1 \rceil$	unchanged	Yes
UI	unchanged	$\lfloor v_1 \rfloor$	Yes

Here v_1 is the value specified by [value1].

CSECTION (optional)

The purpose of the CSECTION is to specify the conic constraint

$$x \in \mathcal{K}$$

in (12.2). It is assumed that \mathcal{K} 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

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

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

- \mathbb{R} set:

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

- Zero cone:

$$\mathcal{K}_t = \{0\} \subseteq \mathbb{R}^{n^t}. \quad (12.3)$$

- Quadratic cone:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\}. \quad (12.4)$$

- Rotated quadratic cone:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : 2x_1x_2 \geq \sum_{j=3}^{n^t} x_j^2, \quad x_1, x_2 \geq 0 \right\}. \quad (12.5)$$

- Primal exponential cone:

$$\mathcal{K}_t = \{x \in \mathbb{R}^3 : x_1 \geq x_2 \exp(x_3/x_2), \quad x_1, x_2 \geq 0\}. \quad (12.6)$$

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

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : x_1^\alpha x_2^{1-\alpha} \geq \sqrt{\sum_{j=3}^{n^t} x_j^2}, \quad x_1, x_2 \geq 0 \right\}. \quad (12.7)$$

- Dual exponential cone:

$$\mathcal{K}_t = \{x \in \mathbb{R}^3 : x_1 \geq -x_3 e^{-1} \exp(x_2/x_3), \quad x_3 \leq 0, x_1 \geq 0\}. \quad (12.8)$$

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

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : \left(\frac{x_1}{\alpha}\right)^\alpha \left(\frac{x_2}{1-\alpha}\right)^{1-\alpha} \geq \sqrt{\sum_{j=3}^{n^t} x_j^2}, \quad x_1, x_2 \geq 0 \right\}. \quad (12.9)$$

In general, membership in the \mathbb{R} set is not specified. If a variable is not a member of any other cone then it is assumed to be a member of the \mathbb{R} cone.

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

$$x_4^{1/3} x_5^{2/3} \geq |x_8|$$

and the rotated quadratic cone

$$2x_3x_7 \geq x_1^2 + x_0^2, \quad x_3, x_7 \geq 0,$$

should be specified in the MPS file. One CSECTION is required for each cone and they are specified as follows:

*	1	2	3	4	5	6
*234567890123456789012345678901234567890123456789012345678901234567890						
CSECTION	konea	3e-1		PPOW		
x4						
x5						
x8						
CSECTION	koneb	0.0		RQUAD		
x7						
x3						
x1						
x0						

In general, a CSECTION header has the format

CSECTION	[kname1]	[value1]	[ktype]
----------	----------	----------	---------

where the requirements for each field are as follows:

Field	Starting Position	Max Width	Required	Description
[kname1]	15	8	Yes	Name of the cone
[value1]	25	12	No	Cone parameter
[ktype]	40		Yes	Type of the cone.

The possible cone type keys are:

[ktype]	Members	[value1]	Interpretation.
ZERO	≥ 0	unused	Zero cone (12.3).
QUAD	≥ 1	unused	Quadratic cone (12.4).
RQUAD	≥ 2	unused	Rotated quadratic cone (12.5).
PEXP	3	unused	Primal exponential cone (12.6).
PPOW	≥ 2	α	Primal power cone (12.7).
DEXP	3	unused	Dual exponential cone (12.8).
DPOW	≥ 2	α	Dual power cone (12.9).

A record in the CSECTION has the format

[vname1]

where the requirements for each field are

Field	Starting Position	Max Width	required	Description
[vname1]	5	8	Yes	A valid variable name

A variable must occur in at most one CSECTION.

ENDATA

This keyword denotes the end of the MPS file.

12.2.2 Integer Variables

Using special bound keys in the **BOUNDS** section it is possible to specify that some or all of the variables should be integer-constrained i.e. be members of \mathcal{J} . However, an alternative method is available. This method is available only for backward compatibility and we recommend that it is not used. This method requires that markers are placed in the **COLUMNS** section as in the example:

COLUMNS				
x1	obj	-10.0	c1	0.7
x1	c2	0.5	c3	1.0
x1	c4	0.1		
* Start of integer-constrained variables.				
MARK000	'MARKER'		'INTORG'	
x2	obj	-9.0	c1	1.0
x2	c2	0.8333333333	c3	0.66666667
x2	c4	0.25		
x3	obj	1.0	c6	2.0
MARK001	'MARKER'		'INTEND'	
* End of integer-constrained variables.				

Please note that special marker lines are used to indicate the start and the end of the integer variables. Furthermore be aware of the following

- All variables between the markers are assigned a default lower bound of 0 and a default upper bound of 1. **This may not be what is intended.** If it is not intended, the correct bounds should be defined in the **BOUNDS** section of the MPS formatted file.
- **MOSEK** ignores field 1, i.e. MARK0001 and MARK001, however, other optimization systems require them.
- Field 2, i.e. **MARKER**, must be specified including the single quotes. This implies that no row can be assigned the name **MARKER**.
- Field 3 is ignored and should be left blank.
- Field 4, i.e. **INTORG** and **INTEND**, must be specified.
- It is possible to specify several such integer marker sections within the **COLUMNS** section.

12.2.3 General Limitations

- An MPS file should be an ASCII file.

12.2.4 Interpretation of the MPS Format

Several issues related to the MPS format are not well-defined by the industry standard. However, **MOSEK** uses the following interpretation:

- If a matrix element in the **COLUMNS** section is specified multiple times, then the multiple entries are added together.
- If a matrix element in a **QSECTION** section is specified multiple times, then the multiple entries are added together.

12.2.5 The Free MPS Format

MOSEK supports a free format variation of the MPS format. The free format is similar to the MPS file format but less restrictive, e.g. it allows longer names. However, a name must not contain any blanks.

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

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

Warning: This file format is to a large extent deprecated. While it can still be used for linear and quadratic problems, for conic problems the [Sec. 12.5](#) is recommended.

12.3 The OPF Format

The *Optimization Problem Format (OPF)* is an alternative to LP and MPS files for specifying optimization problems. It is row-oriented, inspired by the CPLEX LP format.

Apart from containing objective, constraints, bounds etc. it may contain complete or partial solutions, comments and extra information relevant for solving the problem. It is designed to be easily read and modified by hand and to be forward compatible with possible future extensions.

Intended use

The OPF file format is meant to replace several other files:

- The LP file format: Any problem that can be written as an LP file can be written as an OPF file too; furthermore it naturally accommodates ranged constraints and variables as well as arbitrary characters in names, fixed expressions in the objective, empty constraints, and conic constraints.
- Parameter files: It is possible to specify integer, double and string parameters along with the problem (or in a separate OPF file).
- Solution files: It is possible to store a full or a partial solution in an OPF file and later reload it.

12.3.1 The File Format

The format uses tags to structure data. A simple example with the basic sections may look like this:

```
[comment]
This is a comment. You may write almost anything here...
[/comment]

# This is a single-line comment.

[objective min 'myobj']
x + 3 y + x^2 + 3 y^2 + z + 1
[/objective]

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

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

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

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

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

Unnamed arguments are identified by their order, while named arguments may appear in any order, but never before an unnamed argument. The `value` can be a quoted, single-quoted or double-quoted text string, i.e.

```
[tag 'value']      single-quoted value [/tag]
[tag arg='value']  single-quoted value [/tag]
[tag "value"]     double-quoted value [/tag]
[tag arg="value"] double-quoted value [/tag]
```

12.3.2 Sections

The recognized tags are

`[comment]`

A comment section. This can contain *almost* any text: Between single quotes (') or double quotes (") any text may appear. Outside quotes the markup characters ([and]) must be prefixed by backslashes. Both single and double quotes may appear alone or inside a pair of quotes if it is prefixed by a backslash.

`[objective]`

The objective function: This accepts one or two parameters, where the first one (in the above example `min`) is either `min` or `max` (regardless of case) and defines the objective sense, and the second one (above `myobj`), if present, is the objective name. The section may contain linear and quadratic expressions.

If several objectives are specified, all but the last are ignored.

`[constraints]`

This does not directly contain any data, but may contain subsections `con` defining a linear constraint.

`[con]`

Defines a single constraint; if an argument is present (`[con NAME]`) this is used as the name of the constraint, otherwise it is given a null-name. The section contains a constraint definition written as linear and quadratic expressions with a lower bound, an upper bound, with both or with an equality. Examples:

```
[constraints]
[con 'con1'] 0 <= x + y      [/con]
[con 'con2'] 0 >= x + y      [/con]
[con 'con3'] 0 <= x + y <= 10 [/con]
[con 'con4']      x + y = 10 [/con]
[/constraints]
```

Constraint names are unique. If a constraint is specified which has the same name as a previously defined constraint, the new constraint replaces the existing one.

[bounds]

This does not directly contain any data, but may contain subsections **b** (linear bounds on variables) and **cone** (cones).

[b]

Bound definition on one or several variables separated by comma (,). An upper or lower bound on a variable replaces any earlier defined bound on that variable. If only one bound (upper or lower) is given only this bound is replaced. This means that upper and lower bounds can be specified separately. So the OPF bound definition:

<pre>[b] x,y >= -10 [/b] [b] x,y <= 10 [/b]</pre>
--

results in the bound $-10 \leq x, y \leq 10$.

[cone]

Specifies a cone. A cone is defined as a sequence of variables which belong to a single unique cone. The supported cone types are:

- **quad**: a quadratic cone of n variables x_1, \dots, x_n defines a constraint of the form

$$x_1^2 \geq \sum_{i=2}^n x_i^2, \quad x_1 \geq 0.$$

- **rquad**: a rotated quadratic cone of n variables x_1, \dots, x_n defines a constraint of the form

$$2x_1x_2 \geq \sum_{i=3}^n x_i^2, \quad x_1, x_2 \geq 0.$$

- **pexp**: primal exponential cone of 3 variables x_1, x_2, x_3 defines a constraint of the form

$$x_1 \geq x_2 \exp(x_3/x_2), \quad x_1, x_2 \geq 0.$$

- **ppow** with parameter $0 < \alpha < 1$: primal power cone of n variables x_1, \dots, x_n defines a constraint of the form

$$x_1^\alpha x_2^{1-\alpha} \geq \sqrt{\sum_{j=3}^n x_j^2}, \quad x_1, x_2 \geq 0.$$

- **dexp**: dual exponential cone of 3 variables x_1, x_2, x_3 defines a constraint of the form

$$x_1 \geq -x_3 e^{-1} \exp(x_2/x_3), \quad x_3 \leq 0, x_1 \geq 0.$$

- **dpow** with parameter $0 < \alpha < 1$: dual power cone of n variables x_1, \dots, x_n defines a constraint of the form

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

- **zero**: zero cone of n variables x_1, \dots, x_n defines a constraint of the form

$$x_1 = \dots = x_n = 0$$

A [bounds]-section example:

```
[bounds]
[b] 0 <= x,y <= 10 [/b] # ranged bound
[b] 10 >= x,y >= 0 [/b] # ranged bound
[b] 0 <= x,y <= inf [/b] # using inf
[b] x,y free [/b] # free variables
# Let (x,y,z,w) belong to the cone K
[cone rquad] x,y,z,w [/cone] # rotated quadratic cone
[cone ppow '3e-01' 'a'] x1, x2, x3 [/cone] # power cone with alpha=1/3 and name 'a'
[/bounds]
```

By default all variables are free.

[variables]

This defines an ordering of variables as they should appear in the problem. This is simply a space-separated list of variable names.

[integer]

This contains a space-separated list of variables and defines the constraint that the listed variables must be integer-valued.

[hints]

This may contain only non-essential data; for example estimates of the number of variables, constraints and non-zeros. Placed before all other sections containing data this may reduce the time spent reading the file.

In the `hints` section, any subsection which is not recognized by **MOSEK** is simply ignored. In this section a hint is defined as follows:

```
[hint ITEM] value [/hint]
```

The hints recognized by **MOSEK** are:

- `numvar` (number of variables),
- `numcon` (number of linear/quadratic constraints),
- `numanz` (number of linear non-zeros in constraints),
- `numqnz` (number of quadratic non-zeros in constraints).

[solutions]

This section can contain a set of full or partial solutions to a problem. Each solution must be specified using a `[solution]`-section, i.e.

```
[solutions]
[solution]...[/solution] #solution 1
[solution]...[/solution] #solution 2
#other solutions....
[solution]...[/solution] #solution n
[/solutions]
```

The syntax of a `[solution]`-section is the following:

```
[solution SOLTYPE status=STATUS]...[/solution]
```

where `SOLTYPE` is one of the strings

- `interior`, a non-basic solution,
- `basic`, a basic solution,

- **integer**, an integer solution,

and **STATUS** is one of the strings

- UNKNOWN,
- OPTIMAL,
- INTEGER_OPTIMAL,
- PRIM_FEAS,
- DUAL_FEAS,
- PRIM_AND_DUAL_FEAS,
- NEAR_OPTIMAL,
- NEAR_PRIM_FEAS,
- NEAR_DUAL_FEAS,
- NEAR_PRIM_AND_DUAL_FEAS,
- PRIM_INFEAS_CER,
- DUAL_INFEAS_CER,
- NEAR_PRIM_INFEAS_CER,
- NEAR_DUAL_INFEAS_CER,
- NEAR_INTEGER_OPTIMAL.

Most of these values are irrelevant for input solutions; when constructing a solution for simplex hot-start or an initial solution for a mixed integer problem the safe setting is UNKNOWN.

A **[solution]**-section contains **[con]** and **[var]** sections. Each **[con]** and **[var]** section defines solution information for a single variable or constraint, specified as list of **KEYWORD/value** pairs, in any order, written as

KEYWORD=value

Allowed keywords are as follows:

- **sk**. The status of the item, where the **value** is one of the following strings:
 - LOW, the item is on its lower bound.
 - UPR, the item is on its upper bound.
 - FIX, it is a fixed item.
 - BAS, the item is in the basis.
 - SUPBAS, the item is super basic.
 - UNK, the status is unknown.
 - INF, the item is outside its bounds (infeasible).
- **lv1** Defines the level of the item.
- **s1** Defines the level of the dual variable associated with its lower bound.
- **su** Defines the level of the dual variable associated with its upper bound.
- **sn** Defines the level of the variable associated with its cone.
- **y** Defines the level of the corresponding dual variable (for constraints only).

A **[var]** section should always contain the items **sk**, **lv1**, **s1** and **su**. Items **s1** and **su** are not required for **integer** solutions.

A **[con]** section should always contain **sk**, **lv1**, **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    sl=1.5 su=0.0 [/var]

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

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

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

12.3.3 Numbers

Numbers, when used for parameter values or coefficients, are written in the usual way by the **printf** function. That is, they may be prefixed by a sign (+ or -) and may contain an integer part, decimal part and an exponent. The decimal point is always . (a dot). Some examples are

```
1
1.0
.0
1.
1e10
1e+10
1e-10
```

Some *invalid* examples are

```
e10    # invalid, must contain either integer or decimal part
.       # invalid
.e10   # invalid
```

More formally, the following standard regular expression describes numbers as used:

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

12.3.4 Names

Variable names, constraint names and objective name may contain arbitrary characters, which in some cases must be enclosed by quotes (single or double) that in turn must be preceded by a backslash. Unquoted names must begin with a letter (a-z or A-Z) and contain only the following characters: the letters a-z and A-Z, the digits 0-9, braces { and } and underscore (_).

Some examples of legal names:

```
an_unquoted_name
another_name{123}
'single quoted name'
"double quoted name"
"name with \"quote\" in it"
"name with []s in it"
```

12.3.5 Parameters Section

In the `vendor` section solver parameters are defined inside the `parameters` subsection. Each parameter is written as

```
[p PARAMETER_NAME] value [/p]
```

where `PARAMETER_NAME` is replaced by a **MOSEK** parameter name, usually of the form `MSK_IPAR_...`, `MSK_DPAR_...` or `MSK_SPAR_...`, and the `value` is replaced by the value of that parameter; both integer values and named values may be used. Some simple examples are

```
[vendor mosek]
[parameters]
[p MSK_IPAR_OPF_MAX_TERMS_PER_LINE] 10      [/p]
[p MSK_IPAR_OPF_WRITE_PARAMETERS]    MSK_ON [/p]
[p MSK_DPAR_DATA_TOL_BOUND_INF]      1.0e18 [/p]
[/parameters]
[/vendor]
```

12.3.6 Writing OPF Files from MOSEK

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

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

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

12.3.7 Examples

This section contains a set of small examples written in OPF and describing how to formulate linear, quadratic and conic problems.

Linear Example `lo1.opf`

Consider the example:

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

having the bounds

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

In the OPF format the example is displayed as shown in [Listing 12.1](#).

Listing 12.1: Example of an OPF file for a linear problem.

```
[comment]
  The lo1 example in OPF format
[/comment]

[hints]
  [hint NUMVAR] 4 [/hint]
  [hint NUMCON] 3 [/hint]
  [hint NUMANZ] 9 [/hint]
[/hints]

[variables disallow_new_variables]
  x1 x2 x3 x4
[/variables]

[objective maximize 'obj']
  3 x1 + x2 + 5 x3 + x4
[/objective]

[constraints]
  [con 'c1'] 3 x1 +   x2 + 2 x3           = 30 [/con]
  [con 'c2'] 2 x1 +   x2 + 3 x3 +   x4 >= 15 [/con]
  [con 'c3']      2 x2           + 3 x4 <= 25 [/con]
[/constraints]

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

Quadratic Example qo1.opf

An example of a quadratic optimization problem is

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

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

Listing 12.2: Example of an OPF file for a quadratic problem.

```
[comment]
  The qo1 example in OPF format
[/comment]

[hints]
  [hint NUMVAR] 3 [/hint]
  [hint NUMCON] 1 [/hint]
  [hint NUMANZ] 3 [/hint]
  [hint NUMQNZ] 4 [/hint]
[/hints]

[variables disallow_new_variables]
  x1 x2 x3
[/variables]
```

(continues on next page)

```

[objective minimize 'obj']
  # The quadratic terms are often written with a factor of 1/2 as here,
  # but this is not required.

  - x2 + 0.5 ( 2.0 x1 ^ 2 - 2.0 x3 * x1 + 0.2 x2 ^ 2 + 2.0 x3 ^ 2 )
[/objective]

[constraints]
  [con 'c1'] 1.0 <= x1 + x2 + x3 [/con]
[/constraints]

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

```

Conic Quadratic Example cqo1.opf

Consider the example:

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

Please note that the type of the cones is defined by the parameter to `[cone ...]`; the content of the cone-section is the names of variables that belong to the cone. The resulting OPF file is in [Listing 12.3](#).

Listing 12.3: Example of an OPF file for a conic quadratic problem.

```

[comment]
  The cqo1 example in OPF format.
[/comment]

[hints]
  [hint NUMVAR] 6 [/hint]
  [hint NUMCON] 1 [/hint]
  [hint NUMANZ] 3 [/hint]
[/hints]

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

[objective minimize 'obj']
  x4 + x5 + x6
[/objective]

[constraints]
  [con 'c1'] x1 + x2 + 2e+00 x3 = 1e+00 [/con]
[/constraints]

[bounds]
  # We let all variables default to the positive orthant
  [b] 0 <= * [/b]

  # ...and change those that differ from the default

```

(continues on next page)

```

[b] x4,x5,x6 free [/b]

# Define quadratic cone:  $x_4 \geq \sqrt{x_1^2 + x_2^2}$ 
[cone quad 'k1'] x4, x1, x2 [/cone]

# Define rotated quadratic cone:  $2 x_5 x_6 \geq x_3^2$ 
[cone rquad 'k2'] x5, x6, x3 [/cone]
[/bounds]

```

Mixed Integer Example milo1.opf

Consider the mixed integer problem:

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

This can be implemented in OPF with the file in [Listing 12.4](#).

Listing 12.4: Example of an OPF file for a mixed-integer linear problem.

```

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

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

[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]

```

12.4 The CBF Format

This document constitutes the technical reference manual of the *Conic Benchmark Format* with file extension: `.cbf` or `.CBF`. It unifies linear, second-order cone (also known as conic quadratic), exponential cone, power cone and semidefinite optimization with mixed-integer variables. The format has been designed with benchmark libraries in mind, and therefore focuses on compact and easily parsable representations. The CBF format separates problem structure from the problem data.

12.4.1 How Instances Are Specified

This section defines the spectrum of conic optimization problems that can be formulated in terms of the keywords of the CBF format.

In the CBF format, conic optimization problems are considered in the following form:

$$\begin{aligned} & \min / \max && g^{obj} \\ & \text{s.t.} && \begin{aligned} & g_i \in \mathcal{K}_i, & i \in \mathcal{I}, \\ & G_i \in \mathcal{K}_i, & i \in \mathcal{I}^{PSD}, \\ & x_j \in \mathcal{K}_j, & j \in \mathcal{J}, \\ & \overline{X}_j \in \mathcal{K}_j, & j \in \mathcal{J}^{PSD}. \end{aligned} \end{aligned} \tag{12.10}$$

- **Variables** are either scalar variables, x_j for $j \in \mathcal{J}$, or matrix variables, \overline{X}_j for $j \in \mathcal{J}^{PSD}$. Scalar variables can also be declared as integer.
- **Constraints** are affine expressions of the variables, either scalar-valued g_i for $i \in \mathcal{I}$, or matrix-valued G_i for $i \in \mathcal{I}^{PSD}$

$$\begin{aligned} g_i &= \sum_{j \in \mathcal{J}^{PSD}} \langle F_{ij}, X_j \rangle + \sum_{j \in \mathcal{J}} a_{ij} x_j + b_i, \\ G_i &= \sum_{j \in \mathcal{J}} x_j H_{ij} + D_i. \end{aligned}$$

- The **objective function** is a scalar-valued affine expression of the variables, either to be minimized or maximized. We refer to this expression as g^{obj}

$$g^{obj} = \sum_{j \in \mathcal{J}^{PSD}} \langle F_j^{obj}, X_j \rangle + \sum_{j \in \mathcal{J}} a_j^{obj} x_j + b^{obj}.$$

As of version 4 of the format, CBF files can represent the following non-parametric cones \mathcal{K} :

- **Free domain** - A cone in the linear family defined by

$$\{x \in \mathbb{R}^n\}, \text{ for } n \geq 1.$$

- **Positive orthant** - A cone in the linear family defined by

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

- **Negative orthant** - A cone in the linear family defined by

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

- **Fixpoint zero** - A cone in the linear family defined by

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

- **Quadratic cone** - A cone in the second-order cone family defined by

$$\left\{ \begin{pmatrix} p \\ x \end{pmatrix} \in \mathbb{R} \times \mathbb{R}^{n-1}, p^2 \geq x^T x, p \geq 0 \right\}, \text{ for } n \geq 2.$$

- **Rotated quadratic cone** - A cone in the second-order cone family defined by

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

- **Exponential cone** - A cone in the exponential cone family defined by

$$\text{cl}(S_1) = S_1 \cup S_2$$

where,

$$S_1 = \left\{ \begin{pmatrix} t \\ s \\ r \end{pmatrix} \in \mathbb{R}^3, t \geq s e^{\frac{r}{s}}, s \geq 0 \right\}.$$

and,

$$S_2 = \left\{ \begin{pmatrix} t \\ s \\ r \end{pmatrix} \in \mathbb{R}^3, t \geq 0, r \leq 0, s = 0 \right\}.$$

- **Dual Exponential cone** - A cone in the exponential cone family defined by

$$\text{cl}(S_1) = S_1 \cup S_2$$

where,

$$S_1 = \left\{ \begin{pmatrix} t \\ s \\ r \end{pmatrix} \in \mathbb{R}^3, et \geq (-r) e^{\frac{s}{r}}, -r \geq 0 \right\}.$$

and,

$$S_2 = \left\{ \begin{pmatrix} t \\ s \\ r \end{pmatrix} \in \mathbb{R}^3, et \geq 0, s \geq 0, r = 0 \right\}.$$

- **Radial geometric mean cone** - A cone in the power cone family defined by

$$\left\{ \begin{pmatrix} p \\ x \end{pmatrix} \in \mathbb{R}_+^k \times \mathbb{R}^1, \left(\prod_{j=1}^k p_j \right)^{\frac{1}{k}} \geq |x| \right\}, \text{ for } n = k + 1 \geq 2.$$

- **Dual radial geometric mean cone** - A cone in the power cone family defined by

$$\left\{ \begin{pmatrix} p \\ x \end{pmatrix} \in \mathbb{R}_+^k \times \mathbb{R}^1, \left(\prod_{j=1}^k k p_j \right)^{\frac{1}{k}} \geq |x| \right\}, \text{ for } n = k + 1 \geq 2.$$

and, the following parametric cones:

- **Radial power cone** - A cone in the power cone family defined by

$$\left\{ \begin{pmatrix} p \\ x \end{pmatrix} \in \mathbb{R}_+^k \times \mathbb{R}^{n-k}, \left(\prod_{j=1}^k p_j^{\alpha_j} \right)^{\frac{1}{\sigma}} \geq \|x\|_2 \right\}, \text{ for } n \geq k \geq 1.$$

where, $\sigma = \sum_{j=1}^k \alpha_j$ and $\alpha = \mathbb{R}_{++}^k$.

- **Dual radial power cone** - A cone in the power cone family defined by

$$\left\{ \begin{pmatrix} p \\ x \end{pmatrix} \in \mathbb{R}_+^k \times \mathbb{R}^{n-k}, \left(\prod_{j=1}^k \left(\frac{\sigma p_j}{\alpha_j} \right)^{\alpha_j} \right)^{\frac{1}{\sigma}} \geq \|x\|_2 \right\}, \text{ for } n \geq k \geq 1.$$

where, $\sigma = \sum_{j=1}^k \alpha_j$ and $\alpha = \mathbb{R}_{++}^k$.

12.4.2 The Structure of CBF Files

This section defines how information is written in the CBF format, without being specific about the type of information being communicated.

All information items belong to exactly one of the three groups of information. These information groups, and the order they must appear in, are:

1. File format.
2. Problem structure.
3. Problem data.

The first group, file format, provides information on how to interpret the file. The second group, problem structure, provides the information needed to deduce the type and size of the problem instance. Finally, the third group, problem data, specifies the coefficients and constants of the problem instance.

Information items

The format is composed as a list of information items. The first line of an information item is the **KEYWORD**, revealing the type of information provided. The second line - of some keywords only - is the **HEADER**, typically revealing the size of information that follows. The remaining lines are the **BODY** holding the actual information to be specified.

```
KEYWORD
BODY
```

```
KEYWORD
HEADER
BODY
```

The **KEYWORD** determines how each line in the **HEADER** and **BODY** is structured. Moreover, the number of lines in the **BODY** follows either from the **KEYWORD**, the **HEADER**, or from another information item required to precede it.

File encoding and line width restrictions

The format is based on the US-ASCII printable character set with two extensions as listed below. Note, by definition, that none of these extensions can be misinterpreted as printable US-ASCII characters:

- A line feed marks the end of a line, carriage returns are ignored.
- Comment-lines may contain unicode characters in UTF-8 encoding.

The line width is restricted to 512 bytes, with 3 bytes reserved for the potential carriage return, line feed and null-terminator.

Integers and floating point numbers must follow the ISO C decimal string representation in the standard C locale. The format does not impose restrictions on the magnitude of, or number of significant digits in numeric data, but the use of 64-bit integers and 64-bit IEEE 754 floating point numbers should be sufficient to avoid loss of precision.

Comment-line and whitespace rules

The format allows single-line comments respecting the following rule:

- Lines having first byte equal to '#' (US-ASCII 35) are comments, and should be ignored. Comments are only allowed between information items.

Given that a line is not a comment-line, whitespace characters should be handled according to the following rules:

- Leading and trailing whitespace characters should be ignored.
 - The separator between multiple pieces of information on one line, is either one or more whitespace characters.
- Lines containing only whitespace characters are empty, and should be ignored. Empty lines are only allowed between information items.

12.4.3 Problem Specification

The problem structure

The problem structure defines the objective sense, whether it is minimization and maximization. It also defines the index sets, \mathcal{J} , \mathcal{J}^{PSD} , \mathcal{I} and \mathcal{I}^{PSD} , which are all numbered from zero, $\{0, 1, \dots\}$, and empty until explicitly constructed.

- **Scalar variables** are constructed in vectors restricted to a conic domain, such as $(x_0, x_1) \in \mathbb{R}_+^2$, $(x_2, x_3, x_4) \in \mathcal{Q}^3$, etc. In terms of the Cartesian product, this generalizes to

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

which in the CBF format becomes:

```
VAR
n k
K1 n1
K2 n2
...
Kk nk
```

where $\sum_i n_i = n$ is the total number of scalar variables. The list of supported cones is found in Table 12.3. Integrality of scalar variables can be specified afterwards.

- **PSD variables** are constructed one-by-one. That is, $X_j \succeq \mathbf{0}^{n_j \times n_j}$ for $j \in \mathcal{J}^{PSD}$, constructs a matrix-valued variable of size $n_j \times n_j$ restricted to be symmetric positive semidefinite. In the CBF format, this list of constructions becomes:

```

PSDVAR
N
n1
n2
...
nN

```

where N is the total number of PSD variables.

- **Scalar constraints** are constructed in vectors restricted to a conic domain, such as $(g_0, g_1) \in \mathbb{R}_+^2$, $(g_2, g_3, g_4) \in \mathcal{Q}^3$, etc. In terms of the Cartesian product, this generalizes to

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

which in the CBF format becomes:

```

CON
m k
K1 m1
K2 m2
..
Kk mk

```

where $\sum_i m_i = m$ is the total number of scalar constraints. The list of supported cones is found in [Table 12.3](#).

- **PSD constraints** are constructed one-by-one. That is, $G_i \succeq \mathbf{0}^{m_i \times m_i}$ for $i \in \mathcal{I}^{PSD}$, constructs a matrix-valued affine expressions of size $m_i \times m_i$ restricted to be symmetric positive semidefinite. In the CBF format, this list of constructions becomes

```

PSDCON
M
m1
m2
..
mM

```

where M is the total number of PSD constraints.

With the objective sense, variables (with integer indications) and constraints, the definitions of the many affine expressions follow in problem data.

Problem data

The problem data defines the coefficients and constants of the affine expressions of the problem instance. These are considered zero until explicitly defined, implying that instances with no keywords from this information group are, in fact, valid. Duplicating or conflicting information is a failure to comply with the standard. Consequently, two coefficients written to the same position in a matrix (or to transposed positions in a symmetric matrix) is an error.

The affine expressions of the objective, g^{obj} , of the scalar constraints, g_i , and of the PSD constraints, G_i , are defined separately. The following notation uses the standard trace inner product for matrices, $\langle X, Y \rangle = \sum_{i,j} X_{ij} Y_{ij}$.

- The affine expression of the objective is defined as

$$g^{obj} = \sum_{j \in \mathcal{J}^{PSD}} \langle F_j^{obj}, X_j \rangle + \sum_{j \in \mathcal{J}} a_j^{obj} x_j + b^{obj},$$

in terms of the symmetric matrices, F_j^{obj} , and scalars, a_j^{obj} and b^{obj} .

- The affine expressions of the scalar constraints are defined, for $i \in \mathcal{I}$, as

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

in terms of the symmetric matrices, F_{ij} , and scalars, a_{ij} and b_i .

- The affine expressions of the PSD constraints are defined, for $i \in \mathcal{I}^{PSD}$, as

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

in terms of the symmetric matrices, H_{ij} and D_i .

List of cones

The format uses an explicit syntax for symmetric positive semidefinite cones as shown above. For scalar variables and constraints, constructed in vectors, the supported conic domains and their sizes are given as follows.

Table 12.3: Cones available in the CBF format

Name	CBF keyword	Cone family	Cone size
Free domain	F	linear	$n \geq 1$
Positive orthant	L+	linear	$n \geq 1$
Negative orthant	L-	linear	$n \geq 1$
Fixpoint zero	L=	linear	$n \geq 1$
Quadratic cone	Q	second-order	$n \geq 1$
Rotated quadratic cone	QR	second-order	$n \geq 2$
Exponential cone	EXP	exponential	$n = 3$
Dual exponential cone	EXP*	exponential	$n = 3$
Radial geometric mean cone	GMEANABS	power	$n = k + 1 \geq 2$
Dual radial geometric mean cone	GMEANABS*	power	$n = k + 1 \geq 2$
Radial power cone (parametric)	POW	power	$n \geq k \geq 1$
Dual radial power cone (parametric)	POW*	power	$n \geq k \geq 1$

12.4.4 File Format Keywords

VER

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

HEADER: None

BODY: One line formatted as:

INT

This is the version number.

Must appear exactly once in a file, as the first keyword.

POWCONES

Description: Define a lookup table for power cone domains.

HEADER: One line formatted as:

INT INT

This is the number of cones to be specified and the combined length of their dense parameter vectors.

BODY: A list of chunks each specifying the dense parameter vector of a power cone.

CHUNKHEADER: One line formatted as:

INT

This is the parameter vector length.

CHUNKBODY: A list of lines formatted as:

REAL

This is the parameter vector values. The number of lines should match the number stated in the chunk header.

The cone specified at index k (with 0-based indexing) is registered under the CBF name @ k :POW.

POW*CONES

Description: Define a lookup table for dual power cone domains.

HEADER: One line formatted as:

INT INT

This is the number of cones to be specified and the combined length of their dense parameter vectors.

BODY: A list of chunks each specifying the dense parameter vector of a dual power cone.

CHUNKHEADER: One line formatted as:

INT

This is the parameter vector length.

CHUNKBODY: A list of lines formatted as:

REAL

This is the parameter vector values. The number of lines should match the number stated in the chunk header.

The cone specified at index k (with 0-based indexing) is registered under the CBF name @ k :POW*.

OBJSENSE

Description: Define the objective sense.

HEADER: None

BODY: One line formatted as:

STR

having MIN indicates minimize, and MAX indicates maximize. Upper-case letters are required.

Must appear exactly once in a file.

PSDVAR

Description: Construct the PSD variables.

HEADER: One line formatted as:

INT

This is the number of PSD variables in the problem.

BODY: A list of lines formatted as:

INT

This indicates the number of rows (equal to the number of columns) in the matrix-valued PSD variable. The number of lines should match the number stated in the header.

VAR

Description: Construct the scalar variables.

HEADER: One line formatted as:

INT INT

This is the number of scalar variables, followed by the number of conic domains they are restricted to.

BODY: A list of lines formatted as:

STR INT

This indicates the cone name (see [Table 12.3](#)), and the number of scalar variables restricted to this cone. These numbers should add up to the number of scalar variables stated first in the header. The number of lines should match the second number stated in the header.

INT

Description: Declare integer requirements on a selected subset of scalar variables.

HEADER: one line formatted as:

INT

This is the number of integer scalar variables in the problem.

BODY: a list of lines formatted as:

INT

This indicates the scalar variable index $j \in \mathcal{J}$. The number of lines should match the number stated in the header.

Can only be used after the keyword **VAR**.

PSDCON

Description: Construct the PSD constraints.

HEADER: One line formatted as:

INT

This is the number of PSD constraints in the problem.

BODY: A list of lines formatted as:

INT

This indicates the number of rows (equal to the number of columns) in the matrix-valued affine expression of the PSD constraint. The number of lines should match the number stated in the header.

Can only be used after these keywords: **PSDVAR**, **VAR**.

CON

Description: Construct the scalar constraints.

HEADER: One line formatted as:

INT INT

This is the number of scalar constraints, followed by the number of conic domains they restrict to.

BODY: A list of lines formatted as:

STR INT

This indicates the cone name (see [Table 12.3](#)), and the number of affine expressions restricted to this cone. These numbers should add up to the number of scalar constraints stated first in the header. The number of lines should match the second number stated in the header.

Can only be used after these keywords: **PSDVAR**, **VAR**

OBJFCOORD

Description: Input sparse coordinates (quadruplets) to define the symmetric matrices F_j^{obj} , as used in the objective.

HEADER: One line formatted as:

INT

This is the number of coordinates to be specified.

BODY: A list of lines formatted as:

INT INT INT REAL

This indicates the PSD variable index $j \in \mathcal{J}^{PSD}$, the row index, the column index and the coefficient value. The number of lines should match the number stated in the header.

OBJACOORD

Description: Input sparse coordinates (pairs) to define the scalars, a_j^{obj} , as used in the objective.

HEADER: One line formatted as:

INT

This is the number of coordinates to be specified.

BODY: A list of lines formatted as:

INT REAL

This indicates the scalar variable index $j \in \mathcal{J}$ and the coefficient value. The number of lines should match the number stated in the header.

OBJBCOORD

Description: Input the scalar, b^{obj} , as used in the objective.

HEADER: None.

BODY: One line formatted as:

REAL

This indicates the coefficient value.

FCOORD

Description: Input sparse coordinates (quintuplets) to define the symmetric matrices, F_{ij} , as used in the scalar constraints.

HEADER: One line formatted as:

INT

This is the number of coordinates to be specified.

BODY: A list of lines formatted as:

INT INT INT INT REAL

This indicates the scalar constraint index $i \in \mathcal{I}$, the PSD variable index $j \in \mathcal{J}^{PSD}$, the row index, the column index and the coefficient value. The number of lines should match the number stated in the header.

ACCOORD

Description: Input sparse coordinates (triplets) to define the scalars, a_{ij} , as used in the scalar constraints.

HEADER: One line formatted as:

INT

This is the number of coordinates to be specified.

BODY: A list of lines formatted as:

INT INT REAL

This indicates the scalar constraint index $i \in \mathcal{I}$, the scalar variable index $j \in \mathcal{J}$ and the coefficient value. The number of lines should match the number stated in the header.

BCOORD

Description: Input sparse coordinates (pairs) to define the scalars, b_i , as used in the scalar constraints.

HEADER: One line formatted as:

INT

This is the number of coordinates to be specified.

BODY: A list of lines formatted as:

INT REAL

This indicates the scalar constraint index $i \in \mathcal{I}$ and the coefficient value. The number of lines should match the number stated in the header.

HCOORD

Description: Input sparse coordinates (quintuplets) to define the symmetric matrices, H_{ij} , as used in the PSD constraints.

HEADER: One line formatted as:

INT

This is the number of coordinates to be specified.

BODY: A list of lines formatted as

INT INT INT INT REAL

This indicates the PSD constraint index $i \in \mathcal{I}^{PSD}$, the scalar variable index $j \in \mathcal{J}$, the row index, the column index and the coefficient value. The number of lines should match the number stated in the header.

DCOORD

Description: Input sparse coordinates (quadruplets) to define the symmetric matrices, D_i , as used in the PSD constraints.

HEADER: One line formatted as

INT

This is the number of coordinates to be specified.

BODY: A list of lines formatted as:

INT INT INT REAL

This indicates the PSD constraint index $i \in \mathcal{I}^{PSD}$, the row index, the column index and the coefficient value. The number of lines should match the number stated in the header.

12.4.5 CBF Format Examples

Minimal Working Example

The conic optimization problem (12.11) , has three variables in a quadratic cone - first one is integer - and an affine expression in domain 0 (equality constraint).

$$\begin{aligned} & \text{minimize} && 5.1 x_0 \\ & \text{subject to} && 6.2 x_1 + 7.3 x_2 - 8.4 \in \{0\} \\ & && x \in \mathcal{Q}^3, x_0 \in \mathbb{Z}. \end{aligned} \tag{12.11}$$

Its formulation in the Conic Benchmark Format begins with the version of the CBF format used, to safeguard against later revisions.

```
VER
4
```

Next follows the problem structure, consisting of the objective sense, the number and domain of variables, the indices of integer variables, and the number and domain of scalar-valued affine expressions (i.e., the equality constraint).

```
OBJSENSE
MIN

VAR
3 1
Q 3

INT
1
0

CON
1 1
L= 1
```

Finally follows the problem data, consisting of the coefficients of the objective, the coefficients of the constraints, and the constant terms of the constraints. All data is specified on a sparse coordinate form.

```
OBJCOORD
1
0 5.1

ACCOORD
2
0 1 6.2
0 2 7.3

BCCOORD
1
0 -8.4
```

This concludes the example.

Mixing Linear, Second-order and Semidefinite Cones

The conic optimization problem (12.12), has a semidefinite cone, a quadratic cone over unordered subindices, and two equality constraints.

$$\begin{aligned}
 & \text{minimize} && \left\langle \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}, X_1 \right\rangle + x_1 \\
 & \text{subject to} && \left\langle \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, X_1 \right\rangle + x_1 &= 1.0, \\
 & && \left\langle \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, X_1 \right\rangle + x_0 + x_2 &= 0.5, \\
 & && x_1 \geq \sqrt{x_0^2 + x_2^2}, \\
 & && X_1 \succeq \mathbf{0}.
 \end{aligned} \tag{12.12}$$

The equality constraints are easily rewritten to the conic form, $(g_0, g_1) \in \{0\}^2$, by moving constants such that the right-hand-side becomes zero. The quadratic cone does not fit under the **VAR** keyword in this variable permutation. Instead, it takes a scalar constraint $(g_2, g_3, g_4) = (x_1, x_0, x_2) \in \mathcal{Q}^3$, with scalar variables constructed as $(x_0, x_1, x_2) \in \mathbb{R}^3$. Its formulation in the CBF format is reported in the following list

```

# File written using this version of the Conic Benchmark Format:
#       | Version 4.
VER
4

# The sense of the objective is:
#       | Minimize.
OBJSENSE
MIN

# One PSD variable of this size:
#       | Three times three.
PSDVAR
1
3

# Three scalar variables in this one conic domain:
#       | Three are free.
VAR
3 1
F 3

# Five scalar constraints with affine expressions in two conic domains:
#       | Two are fixed to zero.
#       | Three are in conic quadratic domain.
CON
5 2
L= 2
Q 3

# Five coordinates in F~{obj}_j coefficients:
#       | F~{obj}[0][0,0] = 2.0
#       | F~{obj}[0][1,0] = 1.0
#       | and more...
OBJFCOORD
5

```

(continues on next page)

```

0 0 0 2.0
0 1 0 1.0
0 1 1 2.0
0 2 1 1.0
0 2 2 2.0

# One coordinate in a^{obj}_j coefficients:
#       | a^{obj}[1] = 1.0
OBJCOORD
1
1 1.0

# Nine coordinates in F_ij coefficients:
#       | F[0,0][0,0] = 1.0
#       | F[0,0][1,1] = 1.0
#       | and more...
FCOORD
9
0 0 0 0 1.0
0 0 1 1 1.0
0 0 2 2 1.0
1 0 0 0 1.0
1 0 1 0 1.0
1 0 2 0 1.0
1 0 1 1 1.0
1 0 2 1 1.0
1 0 2 2 1.0

# Six coordinates in a_ij coefficients:
#       | a[0,1] = 1.0
#       | a[1,0] = 1.0
#       | and more...
ACCOORD
6
0 1 1.0
1 0 1.0
1 2 1.0
2 1 1.0
3 0 1.0
4 2 1.0

# Two coordinates in b_i coefficients:
#       | b[0] = -1.0
#       | b[1] = -0.5
BCOORD
2
0 -1.0
1 -0.5

```


Mixing Semidefinite Variables and Linear Matrix Inequalities

The standard forms in semidefinite optimization are usually based either on semidefinite variables or linear matrix inequalities. In the CBF format, both forms are supported and can even be mixed as shown.

$$\begin{aligned}
 & \text{minimize} && \left\langle \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, X_1 \right\rangle + x_1 + x_2 + 1 \\
 & \text{subject to} && \left\langle \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, X_1 \right\rangle - x_1 - x_2 && \geq 0.0, \\
 & && x_1 \begin{bmatrix} 0 & 1 \\ 1 & 3 \end{bmatrix} + x_2 \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} && \succeq \mathbf{0}, \\
 & && X_1 && \succeq \mathbf{0}.
 \end{aligned} \tag{12.13}$$

Its formulation in the CBF format is written in what follows

```

# File written using this version of the Conic Benchmark Format:
#   | Version 4.
VER
4

# The sense of the objective is:
#   | Minimize.
OBJSENSE
MIN

# One PSD variable of this size:
#   | Two times two.
PSDVAR
1
2

# Two scalar variables in this one conic domain:
#   | Two are free.
VAR
2 1
F 2

# One PSD constraint of this size:
#   | Two times two.
PSDCON
1
2

# One scalar constraint with an affine expression in this one conic domain:
#   | One is greater than or equal to zero.
CON
1 1
L+ 1

# Two coordinates in F^{obj}_j coefficients:
#   | F^{obj}[0][0,0] = 1.0
#   | F^{obj}[0][1,1] = 1.0
OBJFCOORD
2
0 0 0 1.0
0 1 1 1.0

# Two coordinates in a^{obj}_j coefficients:

```

(continues on next page)

```

#      | a^{obj}[0] = 1.0
#      | a^{obj}[1] = 1.0
OBJCOORD
2
0 1.0
1 1.0

# One coordinate in b^{obj} coefficient:
#      | b^{obj} = 1.0
OBJBCOORD
1.0

# One coordinate in F_{ij} coefficients:
#      | F[0,0][1,0] = 1.0
FCOORD
1
0 0 1 0 1.0

# Two coordinates in a_{ij} coefficients:
#      | a[0,0] = -1.0
#      | a[0,1] = -1.0
ACCOORD
2
0 0 -1.0
0 1 -1.0

# Four coordinates in H_{ij} coefficients:
#      | H[0,0][1,0] = 1.0
#      | H[0,0][1,1] = 3.0
#      | and more...
HCOORD
4
0 0 1 0 1.0
0 0 1 1 3.0
0 1 0 0 3.0
0 1 1 0 1.0

# Two coordinates in D_i coefficients:
#      | D[0][0,0] = -1.0
#      | D[0][1,1] = -1.0
DCCOORD
2
0 0 0 -1.0
0 1 1 -1.0

```

The exponential cone

The conic optimization problem (12.14), has one equality constraint, one quadratic cone constraint and an exponential cone constraint.

$$\begin{aligned} & \text{minimize} && x_0 - x_3 \\ & \text{subject to} && x_0 + 2x_1 - x_2 \in \{0\} \\ & && (5.0, x_0, x_1) \in \mathcal{Q}^3 \\ & && (x_2, 1.0, x_3) \in EXP. \end{aligned} \tag{12.14}$$

The nonlinear conic constraints enforce $\sqrt{x_0^2 + x_1^2} \leq 0.5$ and $x_3 \leq \log(x_2)$.

```
# File written using this version of the Conic Benchmark Format:
#       | Version 3.
VER
3

# The sense of the objective is:
#       | Minimize.
OBJSENSE
MIN

# Four scalar variables in this one conic domain:
#       | Four are free.
VAR
4 1
F 4

# Seven scalar constraints with affine expressions in three conic domains:
#       | One is fixed to zero.
#       | Three are in conic quadratic domain.
#       | Three are in exponential cone domain.
CON
7 3
L= 1
Q 3
EXP 3

# Two coordinates in a^{obj}_j coefficients:
#       | a^{obj}[0] = 1.0
#       | a^{obj}[3] = -1.0
OBJCOORD
2
0 1.0
3 -1.0

# Seven coordinates in a_ij coefficients:
#       | a[0,0] = 1.0
#       | a[0,1] = 2.0
#       | and more...
ACCOORD
7
0 0 1.0
0 1 2.0
0 2 -1.0
2 0 1.0
3 1 1.0
4 2 1.0
6 3 1.0
```

(continues on next page)

(continued from previous page)

```
# Two coordinates in b_i coefficients:
#       | b[1] = 5.0
#       | b[5] = 1.0
BCOORD
2
1 5.0
5 1.0
```

Parametric cones

The problem (12.15), has three variables in a power cone with parameter $\alpha_1 = (1, 1)$ and two power cone constraints each with parameter $\alpha_0 = (8, 1)$.

$$\begin{aligned} & \text{minimize} && x_3 \\ & \text{subject to} && (1.0, x_1, x_1 + x_2) \in POW_{\alpha_0} \\ & && (1.0, x_2, x_1 + x_2) \in POW_{\alpha_0} \\ & && x \in POW_{\alpha_1}. \end{aligned} \tag{12.15}$$

The nonlinear conic constraints enforce $x_3 \leq x_1 x_2$ and $x_1 + x_2 \leq \min(x_1^{\frac{1}{9}}, x_2^{\frac{1}{9}})$.

```
# File written using this version of the Conic Benchmark Format:
#       | Version 3.
VER
3

# Two power cone domains defined in a total of four parameters:
#       | @0:POW (specification 0) has two parameters:
#       | alpha[0] = 8.0.
#       | alpha[1] = 1.0.
#       | @1:POW (specification 1) has two parameters:
#       | alpha[0] = 1.0.
#       | alpha[1] = 1.0.
POWCONES
2 4
2
8.0
1.0
2
1.0
1.0

# The sense of the objective is:
#       | Maximize.
OBJSENSE
MAX

# Three scalar variable in this one conic domain:
#       | Three are in power cone domain (specification 1).
VAR
3 1
@1:POW 3

# Six scalar constraints with affine expressions in two conic domains:
#       | Three are in power cone domain (specification 0).
#       | Three are in power cone domain (specification 0).
```

(continues on next page)

```

CON
6 2
@0:POW 3
@0:POW 3

# One coordinate in a^{obj}_j coefficients:
#       | a^{obj}[2] = 1.0
OBJCOORD
1
2 1.0

# Six coordinates in a_ij coefficients:
#       | a[1,0] = 1.0
#       | a[2,0] = 1.0
#       | and more...
ACCOORD
6
1 0 1.0
2 0 1.0
2 1 1.0
4 1 1.0
5 0 1.0
5 1 1.0

# Two coordinates in b_i coefficients:
#       | b[0] = 1.0
#       | b[3] = 1.0
BCCOORD
2
0 1.0
3 1.0

```

12.5 The PTF Format

The PTF format is a human-readable, natural text format supporting all of **MOSEK** optimization problems in conic form, possibly with integer variables and disjunctive constraints.

12.5.1 The overall format

The format is indentation based, where each section is started by a head line and followed by a section body with deeper indentation than the head line. For example:

```

Header line
  Body line 1
  Body line 1
  Body line 1

```

Section can also be nested:

```

Header line A
  Body line in A
  Header line A.1
    Body line in A.1
    Body line in A.1
  Body line in A

```

The indentation of blank lines is ignored, so a subsection can contain a blank line with no indentation. The character # defines a line comment and anything between the # character and the end of the line is ignored.

In a PTF file, the first section must be a **Task** section. The order of the remaining section is arbitrary, and sections may occur multiple times or not at all.

MOSEK will ignore any top-level section it does not recognize.

Names

In the description of the format we use following definitions for name strings:

```
NAME: PLAIN_NAME | QUOTED_NAME
PLAIN_NAME: [a-zA-Z_] [a-zA-Z0-9_-.!|]
QUOTED_NAME: '"' ( [^'\\r\n] | "\\" ( [\\rn] | "x" [0-9a-fA-F] [0-9a-fA-F] ) ) * "'
```

Expressions

An expression is a sum of terms. A term is either a linear term (a coefficient and a variable name, where the coefficient can be left out if it is 1.0), or a matrix inner product.

An expression:

```
EXPR: EMPTY | [+]? TERM ( [+]? TERM ) *
TERM: LINEAR_TERM | MATRIX_TERM
```

A linear term

```
LINEAR_TERM: FLOAT? NAME
```

A matrix term

```
MATRIX_TERM: "<" FLOAT? NAME ( [+]? FLOAT? NAME ) * ";" NAME ">"
```

Here the right-hand name is the name of a (semidefinite) matrix variable, and the left-hand side is a sum of symmetric matrixes. The actual matrixes are defined in a separate section.

Expressions can span multiple lines by giving subsequent lines a deeper indentation.

For example following two section are equivalent:

```
# Everything on one line:
x1 + x2 + x3 + x4

# Split into multiple lines:
x1
  + x2
  + x3
  + x4
```

12.5.2 Task section

The first section of the file must be a **Task**. The text in this section is not used and may contain comments, or meta-information from the writer or about the content.

Format:

```
Task NAME
  Anything goes here...
```

NAME is a the task name.

12.5.3 Objective section

The **Objective** section defines the objective name, sense and function. The format:

```
"Objective" NAME?  
  ( "Minimize" | "Maximize" ) EXPR
```

For example:

```
Objective 'obj'  
  Minimize x1 + 0.2 x2 + < M1 ; X1 >
```

12.5.4 Constraints section

The constraints section defines a series of constraints. A constraint defines a term $A \cdot x + b \in K$. For linear constraints A is just one row, while for conic constraints it can be multiple rows. If a constraint spans multiple rows these can either be written inline separated by semi-colons, or each expression in a separate sub-section.

Simple linear constraints:

```
"Constraints"  
  NAME? "[" [-+] (FLOAT | "Inf") (";" [-+] (FLOAT | "Inf"))? "]" EXPR
```

If the brackets contain two values, they are used as upper and lower bounds. If they contain one value the constraint is an equality.

For example:

```
Constraints  
'c1' [0;10] x1 + x2 + x3  
[0] x1 + x2 + x3
```

Constraint blocks put the expression either in a subsection or inline. The cone type (domain) is written in the brackets, and **MOSEK** currently supports following types:

- SOC(N) Second order cone of dimension N
- RSOC(N) Rotated second order cone of dimension N
- PSD(N) Symmetric positive semidefinite cone of dimension N. This contains $N \cdot (N+1) / 2$ elements.
- PEXP Primal exponential cone of dimension 3
- DEXP Dual exponential cone of dimension 3
- PPOW(N,P) Primal power cone of dimension N with parameter P
- DPOW(N,P) Dual power cone of dimension N with parameter P
- ZERO(N) The zero-cone of dimension N.

```
"Constraints"  
  NAME? "[" DOMAIN "]" EXPR_LIST
```

For example:

```
Constraints  
'K1' [SOC(3)] x1 + x2 ; x2 + x3 ; x3 + x1  
'K2' [RSOC(3)]  
  x1 + x2  
  x2 + x3  
  x3 + x1
```

12.5.5 Variables section

Any variable used in an expression must be defined in a variable section. The variable section defines each variable domain.

```
"Variables"
  NAME "[" [-+] (FLOAT | "Inf") (";" [-+] (FLOAT | "Inf"))? "]"
  NAME "[" DOMAIN "]" NAMES
```

For example, a linear variable

```
Variables
  x1 [0;Inf]
```

As with constraints, members of a conic domain can be listed either inline or in a subsection:

```
Variables
  k1 [SOC(3)] x1 ; x2 ; x3
  k2 [RSOC(3)]
    x1
    x2
    x3
```

12.5.6 Integer section

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

```
Integer
  x1 x2 x3
```

12.5.7 SymmetricMatrixes section

This section defines the symmetric matrixes used for matrix coefficients in matrix inner product terms. The section lists named matrixes, each with a size and a number of non-zeros. Only non-zeros in the lower triangular part should be defined.

```
"SymmetricMatrixes"
  NAME "SYMMAT" "(" INT ")" ( "(" INT "," INT "," FLOAT ")" ) *
  ...
```

For example:

```
SymmetricMatrixes
  M1 SYMMAT(3) (0,0,1.0) (1,1,2.0) (2,1,0.5)
  M2 SYMMAT(3)
    (0,0,1.0)
    (1,1,2.0)
    (2,1,0.5)
```


12.5.8 Solutions section

Each subsection defines a solution. A solution defines for each constraint and for each variable exactly one primal value and either one (for conic domains) or two (for linear domains) dual values. The values follow the same logic as in the **MOSEK C API**. A primal and a dual solution status defines the meaning of the values primal and dual (solution, certificate, unknown, etc.)

The format is this:

```
"Solutions"
  "Solution" WHICHSOL
    "ProblemStatus" PROSTA PROSTA?
  "SolutionStatus" SOLSTA SOLSTA?
  "Objective" FLOAT FLOAT
  "Variables"
    # Linear variable status: level, slx, sux
    NAME "[" STATUS "]" FLOAT (FLOAT FLOAT)?
    # Conic variable status: level, snx
    NAME
      "[" STATUS "]" FLOAT FLOAT?
    ...
  "Constraints"
    # Linear variable status: level, slx, sux
    NAME "[" STATUS "]" FLOAT (FLOAT FLOAT)?
    # Conic variable status: level, snx
    NAME
      "[" STATUS "]" FLOAT FLOAT?
    ...
```

Following values for **WHICHSOL** are supported:

- **interior** Interior solution, the result of an interior-point solver.
- **basic** Basic solution, as produced by a simplex solver.
- **integer** Integer solution, the solution to a mixed-integer problem. This does not define a dual solution.

Following values for **PROSTA** are supported:

- **unknown** The problem status is unknown
- **feasible** The problem has been proven feasible
- **infeasible** The problem has been proven infeasible
- **illposed** The problem has been proved to be ill posed
- **infeasible_or_unbounded** The problem is infeasible or unbounded

Following values for **SOLSTA** are supported:

- **unknown** The solution status is unknown
- **feasible** The solution is feasible
- **optimal** The solution is optimal
- **infeas_cert** The solution is a certificate of infeasibility
- **illposed_cert** The solution is a certificate of illposedness

Following values for **STATUS** are supported:

- **unknown** The value is unknown
- **super_basic** The value is super basic

- `at_lower` The value is basic and at its lower bound
- `at_upper` The value is basic and at its upper bound
- `fixed` The value is basic fixed
- `infinite` The value is at infinity

12.5.9 Examples

Linear example `lo1.ptf`

```
Task ''
# Written by MOSEK v10.0.13
# problemtype: Linear Problem
# number of linear variables: 4
# number of linear constraints: 3
# number of old-style A nonzeros: 9
Objective obj
  Maximize + 3 x1 + x2 + 5 x3 + x4
Constraints
  c1 [3e+1] + 3 x1 + x2 + 2 x3
  c2 [1.5e+1;+inf] + 2 x1 + x2 + 3 x3 + x4
  c3 [-inf;2.5e+1] + 2 x2 + 3 x4
Variables
  x1 [0;+inf]
  x2 [0;1e+1]
  x3 [0;+inf]
  x4 [0;+inf]
```

Conic example `cq01.ptf`

```
Task ''
# Written by MOSEK v10.0.17
# problemtype: Conic Problem
# number of linear variables: 6
# number of linear constraints: 1
# number of old-style cones: 0
# number of positive semidefinite variables: 0
# number of positive semidefinite matrixes: 0
# number of affine conic constraints: 2
# number of disjunctive constraints: 0
# number scalar affine expressions/nonzeros : 6/6
# number of old-style A nonzeros: 3
Objective obj
  Minimize + x4 + x5 + x6
Constraints
  c1 [1] + x1 + x2 + 2 x3
  k1 [QUAD(3)]
    @ac1: + x4
    @ac2: + x1
    @ac3: + x2
  k2 [RQUAD(3)]
    @ac4: + x5
    @ac5: + x6
    @ac6: + x3
Variables
```

(continues on next page)

```

x4
x1 [0;+inf]
x2 [0;+inf]
x5
x6
x3 [0;+inf]

```

Disjunctive example djc1.ptf

```

Task djc1
Objective ''
  Minimize + 2 'x[0]' + 'x[1]' + 3 'x[2]' + 'x[3]'
Constraints
  @c0 [-10;+inf] + 'x[0]' + 'x[1]' + 'x[2]' + 'x[3]'
  @D0 [OR]
    [AND]
      [NEGATIVE(1)]
        + 'x[0]' - 2 'x[1]' + 1
      [ZERO(2)]
        + 'x[2]'
        + 'x[3]'
    [AND]
      [NEGATIVE(1)]
        + 'x[2]' - 3 'x[3]' + 2
      [ZERO(2)]
        + 'x[0]'
        + 'x[1]'
  @D1 [OR]
    [ZERO(1)]
      + 'x[0]' - 2.5
    [ZERO(1)]
      + 'x[1]' - 2.5
    [ZERO(1)]
      + 'x[2]' - 2.5
    [ZERO(1)]
      + 'x[3]' - 2.5
Variables
  'x[0]'
  'x[1]'
  'x[2]'
  'x[3]'

```

12.6 The Task Format

The Task format is **MOSEK**'s native binary format. It contains a complete image of a **MOSEK** task, i.e.

- Problem data: Linear, conic, semidefinite and quadratic data
- Problem item names: Variable names, constraints names, cone names etc.
- Parameter settings
- Solutions

There are a few things to be aware of:

- Status of a solution read from a file will *always* be unknown.
- Parameter settings in a task file *always override* any parameters set on the command line or in a parameter file.

The format is based on the *TAR* (USTar) file format. This means that the individual pieces of data in a `.task` file can be examined by unpacking it as a *TAR* file. Please note that the inverse may not work: Creating a file using *TAR* will most probably not create a valid **MOSEK** Task file since the order of the entries is important.

12.7 The JSON Format

MOSEK provides the possibility to read/write problems and solutions in JSON format. The official JSON website <http://www.json.org> provides plenty of information along with the format definition. JSON is an industry standard for data exchange and JSON files can be easily written and read in most programming languages using dedicated libraries.

MOSEK uses two JSON-based formats:

- **JTASK**, for storing problem instances together with solutions and parameters. The JTASK format contains the same information as a native **MOSEK** task *task format*, that is a very close representation of the internal data storage in the task object.

You can write a JTASK file specifying the extension `.jtask`. When the parameter `MSK_IPAR_WRITE_JSON_INDENTATION` is set the JTASK file will be indented to slightly improve readability.

- **JSOL**, for storing solutions and information items.

You can write a JSOL solution file using the option `-jsolo`. When the parameter `MSK_IPAR_WRITE_JSON_INDENTATION` is set the JSOL file will be indented to slightly improve readability.

You can read a JSOL solution into an existing task file using the option `-jsoli`. Only the `Task/solutions` section of the data will be taken into consideration.

12.7.1 JTASK Specification

The JTASK is a dictionary containing the following sections. All sections are optional and can be omitted if irrelevant for the problem.

- `$schema`: JSON schema.
- `Task/name`: The name of the task (string).
- `Task/INFO`: Information about problem data dimensions and similar. These are treated as hints when reading the file.
 - `numvar`: number of variables (int32).
 - `numcon`: number of constraints (int32).
 - `numcone`: number of cones (int32, deprecated).
 - `numbarvar`: number of symmetric matrix variables (int32).
 - `numanz`: number of nonzeros in A (int64).
 - `numsymmat`: number of matrices in the symmetric matrix storage E (int64).
 - `numafe`: number of affine expressions in AFE storage (int64).
 - `numfnz`: number of nonzeros in F (int64).
 - `numacc`: number of affine conic constraints (ACCs) (int64).
 - `numdjc`: number of disjunctive constraints (DJCs) (int64).
 - `numdom`: number of domains (int64).
 - `mosekver`: MOSEK version (list(int32)).

- **Task/data:** Numerical and structural data of the problem.
 - **var:** Information about variables. All fields present must have the same length as **bk**. All or none of **bk**, **bl**, and **bu** must appear.
 - * **name:** Variable names (list(string)).
 - * **bk:** Bound keys (list(string)).
 - * **bl:** Lower bounds (list(double)).
 - * **bu:** Upper bounds (list(double)).
 - * **type:** Variable types (list(string)).
 - **con:** Information about linear constraints. All fields present must have the same length as **bk**. All or none of **bk**, **bl**, and **bu** must appear.
 - * **name:** Constraint names (list(string)).
 - * **bk:** Bound keys (list(string)).
 - * **bl:** Lower bounds (list(double)).
 - * **bu:** Upper bounds (list(double)).
 - **barvar:** Information about symmetric matrix variables. All fields present must have the same length as **dim**.
 - * **name:** Barvar names (list(string)).
 - * **dim:** Dimensions (list(int32)).
 - **objective:** Information about the objective.
 - * **name:** Objective name (string).
 - * **sense:** Objective sense (string).
 - * **c:** The linear part c of the objective as a sparse vector. Both arrays must have the same length.
 - **subj:** indices of nonzeros (list(int32)).
 - **val:** values of nonzeros (list(double)).
 - * **cfix:** Constant term in the objective (double).
 - * **Q:** The quadratic part Q^o of the objective as a sparse matrix, only lower-triangular part included. All arrays must have the same length.
 - **subi:** row indices of nonzeros (list(int32)).
 - **subj:** column indices of nonzeros (list(int32)).
 - **val:** values of nonzeros (list(double)).
 - * **barc:** The semidefinite part \overline{C} of the objective (list). Each element of the list is a list describing one entry \overline{C}_j using three fields:
 - index j (int32).
 - weights of the matrices from the storage E forming \overline{C}_j (list(double)).
 - indices of the matrices from the storage E forming \overline{C}_j (list(int64)).
 - **A:** The linear constraint matrix A as a sparse matrix. All arrays must have the same length.
 - * **subi:** row indices of nonzeros (list(int32)).
 - * **subj:** column indices of nonzeros (list(int32)).
 - * **val:** values of nonzeros (list(double)).
 - **bara:** The semidefinite part \overline{A} of the constraints (list). Each element of the list is a list describing one entry \overline{A}_{ij} using four fields:
 - * index i (int32).
 - * index j (int32).
 - * weights of the matrices from the storage E forming \overline{A}_{ij} (list(double)).
 - * indices of the matrices from the storage E forming \overline{A}_{ij} (list(int64)).
 - **AFE:** The affine expression storage.
 - * **numafe:** number of rows in the storage (int64).
 - * **F:** The matrix F as a sparse matrix. All arrays must have the same length.

- **subi**: row indices of nonzeros (list(int64)).
 - **subj**: column indices of nonzeros (list(int32)).
 - **val**: values of nonzeros (list(double)).
- * **g**: The vector g of constant terms as a sparse vector. Both arrays must have the same length.
 - **subi**: indices of nonzeros (list(int64)).
 - **val**: values of nonzeros (list(double)).
- * **barf**: The semidefinite part \bar{F} of the expressions in AFE storage (list). Each element of the list is a list describing one entry \bar{F}_{ij} using four fields:
 - index i (int64).
 - index j (int32).
 - weights of the matrices from the storage E forming \bar{F}_{ij} (list(double)).
 - indices of the matrices from the storage E forming \bar{F}_{ij} (list(int64)).
- **domains**: Information about domains. All fields present must have the same length as **type**.
 - * **name**: Domain names (list(string)).
 - * **type**: Description of the type of each domain (list). Each element of the list is a list describing one domain using at least one field:
 - domain type (string).
 - (except **pexp**, **dexp**) dimension (int64).
 - (only **ppow**, **dpow**) weights (list(double)).
- **ACC**: Information about affine conic constraints (ACC). All fields present must have the same length as **domain**.
 - * **name**: ACC names (list(string)).
 - * **domain**: Domains (list(int64)).
 - * **afeidx**: AFE indices, grouped by ACC (list(list(int64))).
 - * **b**: constant vectors b , grouped by ACC (list(list(double))).
- **DJC**: Information about disjunctive constraints (DJC). All fields present must have the same length as **termsize**.
 - * **name**: DJC names (list(string)).
 - * **termsize**: Term sizes, grouped by DJC (list(list(int64))).
 - * **domain**: Domains, grouped by DJC (list(list(int64))).
 - * **afeidx**: AFE indices, grouped by DJC (list(list(int64))).
 - * **b**: constant vectors b , grouped by DJC (list(list(double))).
- **MatrixStore**: The symmetric matrix storage E (list). Each element of the list is a list describing one entry E using four fields in sparse matrix format, lower-triangular part only:
 - * dimension (int32).
 - * row indices of nonzeros (list(int32)).
 - * column indices of nonzeros (list(int32)).
 - * values of nonzeros (list(double)).
- **Q**: The quadratic part Q^c of the constraints (list). Each element of the list is a list describing one entry Q_i^c using four fields in sparse matrix format, lower-triangular part only:
 - * the row index i (int32).
 - * row indices of nonzeros (list(int32)).
 - * column indices of nonzeros (list(int32)).
 - * values of nonzeros (list(double)).
- **qcone** (deprecated). The description of cones. All fields present must have the same length as **type**.
 - * **name**: Cone names (list(string)).
 - * **type**: Cone types (list(string)).
 - * **par**: Additional cone parameters (list(double)).

* **members**: Members, grouped by cone (list(list(int32))).

- **Task/solutions**: Solutions. This section can contain up to three subsections called:

- **interior**
- **basic**
- **integer**

corresponding to the three solution types in MOSEK. Each of these sections has the same structure:

- **prosta**: problem status (string).
- **solsta**: solution status (string).
- **xx, xc, y, slc, suc, slx, sux, snx**: one for each component of the solution of the same name (list(double)).
- **skx, skc, skn**: status keys (list(string)).
- **doty**: the dual y solution, grouped by ACC (list(list(double))).
- **barx, bars**: the primal/dual semidefinite solution, grouped by matrix variable (list(list(double))).

- **Task/parameters**: Parameters.

- **iparam**: Integer parameters (dictionary). A dictionary with entries of the form **name:value**, where **name** is a shortened parameter name (without leading **MSK_IPAR_**) and **value** is either an integer or string if the parameter takes values from an enum.
- **dparam**: Double parameters (dictionary). A dictionary with entries of the form **name:value**, where **name** is a shortened parameter name (without leading **MSK_DPAR_**) and **value** is a double.
- **sparam**: String parameters (dictionary). A dictionary with entries of the form **name:value**, where **name** is a shortened parameter name (without leading **MSK_SPAR_**) and **value** is a string. Note that this section is allowed but MOSEK ignores it both when writing and reading JTASK files.

12.7.2 JSOL Specification

The JSOL is a dictionary containing the following sections. All sections are optional and can be omitted if irrelevant for the problem.

- **\$schema**: JSON schema.
- **Task/name**: The name of the task (string).
- **Task/solutions**: Solutions. This section can contain up to three subsections called:

- **interior**
- **basic**
- **integer**

corresponding to the three solution types in MOSEK. Each of these section has the same structure:

- **prosta**: problem status (string).
- **solsta**: solution status (string).
- **xx, xc, y, slc, suc, slx, sux, snx**: one for each component of the solution of the same name (list(double)).
- **skx, skc, skn**: status keys (list(string)).
- **doty**: the dual y solution, grouped by ACC (list(list(double))).
- **barx, bars**: the primal/dual semidefinite solution, grouped by matrix variable (list(list(double))).

- Task/information: Information items from the optimizer.
 - int32: int32 information items (dictionary). A dictionary with entries of the form **name**: **value**.
 - int64: int64 information items (dictionary). A dictionary with entries of the form **name**: **value**.
 - double: double information items (dictionary). A dictionary with entries of the form **name**: **value**.

12.7.3 A jtask example

Listing 12.5: A formatted jtask file for a simple portfolio optimization problem.

```
{
  "$schema": "http://mosek.com/json/schema#",
  "Task/name": "Markowitz portfolio with market impact",
  "Task/INFO": {"numvar": 7, "numcon": 1, "numcone": 0, "numbarvar": 0, "numanz": 6, "numsymmat"
  ↪ ": 0, "numafe": 13, "numfnz": 12, "numacc": 4, "numdjc": 0, "numdom": 3, "mosekver": [10, 0, 0, 3]},
  "Task/data": {
    "var": {
      "name": ["1.0", "x[0]", "x[1]", "x[2]", "t[0]", "t[1]", "t[2]"],
      "bk": ["fx", "lo", "lo", "lo", "fr", "fr", "fr"],
      "bl": [1, 0.0, 0.0, 0.0, -1e+30, -1e+30, -1e+30],
      "bu": [1, 1e+30, 1e+30, 1e+30, 1e+30, 1e+30, 1e+30],
      "type": ["cont", "cont", "cont", "cont", "cont", "cont", "cont"]
    },
    "con": {
      "name": ["budget[]"],
      "bk": ["fx"],
      "bl": [1],
      "bu": [1]
    },
    "objective": {
      "sense": "max",
      "name": "obj",
      "c": {
        "subj": [1, 2, 3],
        "val": [0.1073, 0.0737, 0.0627]
      },
      "cfix": 0.0
    },
    "A": {
      "subi": [0, 0, 0, 0, 0, 0],
      "subj": [1, 2, 3, 4, 5, 6],
      "val": [1, 1, 1, 0.01, 0.01, 0.01]
    },
    "AFE": {
      "numafe": 13,
      "F": {
        "subi": [1, 1, 1, 2, 2, 3, 4, 6, 7, 9, 10, 12],
        "subj": [1, 2, 3, 2, 3, 3, 4, 1, 5, 2, 6, 3],
        "val": [0.166673333200005, 0.0232190712557243, 0.0012599496030238, 0.
        ↪ 102863378954911, -0.00222873156550421, 0.0338148677744977, 1, 1, 1, 1, 1, 1]
      },
      "g": {
        "subi": [0, 5, 8, 11],
```

(continues on next page)


```

        "val": [0.035, 1, 1, 1]
    },
    "domains": {
        "type": [
            ["r", 0],
            ["quad", 4],
            ["ppow", 3, [0.6666666666666666, 0.3333333333333337]]
        ],
    },
    "ACC": {
        "name": ["risk[]", "tz[0]", "tz[1]", "tz[2]"],
        "domain": [1, 2, 2, 2],
        "afeidx": [
            [0, 1, 2, 3],
            [4, 5, 6],
            [7, 8, 9],
            [10, 11, 12]
        ]
    },
},
"Task/solutions": {
    "interior": {
        "prosta": "unknown",
        "solsta": "unknown",
        "skx": ["fix", "supbas", "supbas", "supbas", "supbas", "supbas", "supbas"],
        "skc": ["fix"],
        "xx": [1, 0.10331580274282556, 0.11673185566457132, 0.7724326587076371, 0.
↪ 033208600335718846, 0.03988270849469869, 0.6788769587942524],
        "xc": [1],
        "slx": [0.0, -5.585840467641202e-10, -8.945844685006369e-10, -7.815248786428623e-
↪ 11, 0.0, 0.0, 0.0],
        "sux": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        "snx": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        "slc": [0.0],
        "suc": [-0.046725814048521205],
        "y": [0.046725814048521205],
        "doty": [
            [-0.6062603164682975, 0.3620818321879349, 0.17817754087278295, 0.
↪ 4524390346223723],
            [-4.6725842015519993e-4, -7.708781121860897e-6, 2.24800624747081e-4],
            [-4.6725842015519993e-4, -9.268264309496919e-6, 2.390390600079771e-4],
            [-4.6725842015519993e-4, -1.5854982159992136e-4, 6.159249331148646e-4]]
        ]
    },
},
"Task/parameters": {
    "iparam": {
        "LICENSE_DEBUG": "ON",
        "MIO_SEED": 422
    },
    "dparam": {
        "MIO_MAX_TIME": 100
    },
    "sparam": {
    }
}
}

```

12.8 The Solution File Format

MOSEK can output solutions to a text file:

- *basis solution file* (extension `.bas`) if the problem is optimized using the simplex optimizer or basis identification is performed,
- *interior solution file* (extension `.sol`) if a problem is optimized using the interior-point optimizer and no basis identification is required,
- *integer solution file* (extension `.int`) if the problem is solved with the mixed-integer optimizer.

All solution files have the format:

NAME	:	<problem name>					
PROBLEM STATUS	:	<status of the problem>					
SOLUTION STATUS	:	<status of the solution>					
OBJECTIVE NAME	:	<name of the objective function>					
PRIMAL OBJECTIVE	:	<primal objective value corresponding to the solution>					
DUAL OBJECTIVE	:	<dual objective value corresponding to the solution>					
CONSTRAINTS							
INDEX	NAME	AT	ACTIVITY	LOWER LIMIT	UPPER LIMIT	DUAL LOWER	DUAL UPPER
?	<name>	??	<a value>	<a value>	<a value>	<a value>	<a value>
AFFINE CONIC CONSTRAINTS							
INDEX	NAME	I	ACTIVITY	DUAL			
?	<name>	<a value>	<a value>	<a value>			
VARIABLES							
INDEX	NAME	AT	ACTIVITY	LOWER LIMIT	UPPER LIMIT	DUAL LOWER	DUAL UPPER
↪[CONIC DUAL]							
?	<name>	??	<a value>	<a value>	<a value>	<a value>	<a value>
↪[<a value>]							
SYMMETRIC MATRIX VARIABLES							
INDEX	NAME	I	J	PRIMAL	DUAL		
?	<name>	<a value>	<a value>	<a value>	<a value>		

The fields `?`, `??` and `<>` will be filled with problem and solution specific information as described below. The solution contains sections corresponding to parts of the input. Empty sections may be omitted and fields in `[]` are optional, depending on what type of problem is solved. The notation below follows the **MOSEK** naming convention for parts of the solution as defined in the problem specifications in [Sec. 8](#).

- **HEADER** In this section, first the name of the problem is listed and afterwards the problem and solution status are shown. Next the primal and dual objective values are displayed.
- **CONSTRAINTS**
 - **INDEX**: A sequential index assigned to the constraint by **MOSEK**
 - **NAME**: The name of the constraint assigned by the user or autogenerated.
 - **AT**: The status key `bkc` of the constraint as in [Table 12.4](#).
 - **ACTIVITY**: the activity `xc` of the constraint expression.
 - **LOWER LIMIT**: the lower bound `blc` of the constraint.
 - **UPPER LIMIT**: the upper bound `buc` of the constraint.
 - **DUAL LOWER**: the dual multiplier `slc` corresponding to the lower limit on the constraint.
 - **DUAL UPPER**: the dual multiplier `suc` corresponding to the upper limit on the constraint.
- **AFFINE CONIC CONSTRAINTS**

- INDEX: A sequential index assigned to the affine expressions by **MOSEK**
- NAME: The name of the affine conic constraint assigned by the user or autogenerated.
- I: The sequential index of the affine expression in the affine conic constraint.
- ACTIVITY: the activity of the I-th affine expression in the affine conic constraint.
- DUAL: the dual multiplier `doty` for the I-th entry in the affine conic constraint.

- VARIABLES

- INDEX: A sequential index assigned to the variable by **MOSEK**
- NAME: The name of the variable assigned by the user or autogenerated.
- AT: The status key `bkx` of the variable as in Table 12.4.
- ACTIVITY: the value `xx` of the variable.
- LOWER LIMIT: the lower bound `blx` of the variable.
- UPPER LIMIT: the upper bound `bux` of the variable.
- DUAL LOWER: the dual multiplier `slx` corresponding to the lower limit on the variable.
- DUAL UPPER: the dual multiplier `sux` corresponding to the upper limit on the variable.
- CONIC DUAL: the dual multiplier `skx` corresponding to a conic variable (deprecated).

- SYMMETRIC MATRIX VARIABLES

- INDEX: A sequential index assigned to each symmetric matrix entry by **MOSEK**
- NAME: The name of the symmetric matrix variable assigned by the user or autogenerated.
- I: The row index in the symmetric matrix variable.
- J: The column index in the symmetric matrix variable.
- PRIMAL: the value of `barx` for the (I, J)-th entry in the symmetric matrix variable.
- DUAL: the dual multiplier `bars` for the (I, J)-th entry in the symmetric matrix variable.

Table 12.4: Status keys.

Status key	Interpretation
UN	Unknown status
BS	Is basic
SB	Is superbasic
LL	Is at the lower limit (bound)
UL	Is at the upper limit (bound)
EQ	Lower limit is identical to upper limit
**	Is infeasible i.e. the lower limit is greater than the upper limit.

Example.

Below is an example of a solution file.

Listing 12.6: An example of `.sol` file.

NAME	:			
PROBLEM STATUS	:	PRIMAL_AND_DUAL_FEASIBLE		
SOLUTION STATUS	:	OPTIMAL		
OBJECTIVE NAME	:	OBJ		
PRIMAL OBJECTIVE	:	0.70571049347734		
DUAL OBJECTIVE	:	0.70571048919757		
CONSTRAINTS				
INDEX	NAME	AT	ACTIVITY	LOWER LIMIT
				UPPER LIMIT
↪	DUAL LOWER		DUAL UPPER	

(continues on next page)

(continued from previous page)

AFFINE CONIC CONSTRAINTS

INDEX	NAME	I	ACTIVITY	DUAL
0	A1	0	1.0000000009656	0.54475821296644
1	A1	1	0.50000000152223	0.32190455246225
2	A2	0	0.25439922724695	0.4552417870329
3	A2	1	0.17988741850378	-0.32190455246178
4	A2	2	0.17988741850378	-0.32190455246178

VARIABLES

INDEX	NAME	AT	ACTIVITY	LOWER LIMIT	UPPER LIMIT
↪	DUAL LOWER		DUAL UPPER		
0	X1	SB	0.25439922724695	NONE	NONE
↪	0		0		
1	X2	SB	0.17988741850378	NONE	NONE
↪	0		0		
2	X3	SB	0.17988741850378	NONE	NONE
↪	0		0		

SYMMETRIC MATRIX VARIABLES

INDEX	NAME	I	J	PRIMAL	DUAL
0	BARX1	0	0	0.21725733689874	1.1333372337141
1	BARX1	1	0	-0.25997257078534	0.
↪	67809544651396				
2	BARX1	2	0	0.21725733648507	-0.
↪	3219045527104				
3	BARX1	1	1	0.31108610088839	1.1333372332693
4	BARX1	2	1	-0.25997257078534	0.
↪	67809544651435				
5	BARX1	2	2	0.21725733689874	1.1333372337145
6	BARX2	0	0	4.8362272828127e-10	0.
↪	54475821339698				
7	BARX2	1	0	0	0
8	BARX2	1	1	4.8362272828127e-10	0.
↪	54475821339698				

Chapter 13

List of examples

List of examples shipped in the distribution of Command Line Tools:

Table 13.1: List of distributed examples

File	Description
25fv47.mps	A large linear problem from the Netlib library
ampl1.res	Interfacing MOSEK from AMPL
ampl2.res	Interfacing MOSEK from AMPL
ampl3.res	Interfacing MOSEK from AMPL
cqo1.jtask	A simple conic quadratic problem
cqo1.mps	A simple conic quadratic problem
cqo1.ptf	A simple conic quadratic problem
diet.dat	Data for the diet example <code>diet.mod</code>
diet.mod	A diet balancing AMPL example
dinfeas.lp	A simple dual infeasible linear problem
djc1.jtask	A simple problem with disjunctive constraints (DJC)
djc1.ptf	A simple problem with disjunctive constraints (DJC)
feasrepair.lp	An example demonstrating repair of infeasible problems
infeas.lp	A simple primal infeasible problem
lo1.jtask	A simple linear problem
lo1.lp	A simple linear problem
lo1.mps	A simple linear problem
lo1.ptf	A simple linear problem
milol1.jtask	A simple mixed-integer linear problem
milol1.lp	A simple mixed-integer linear problem
milol1.ptf	A simple mixed-integer linear problem
qo1.jtask	A simple quadratic problem
qo1.mps	A simple quadratic problem
qo1.opf	A simple quadratic problem
sdo1.cbf	A simple semidefinite problem with one matrix variable and a quadratic cone
sdo1.jtask	A simple semidefinite problem with one matrix variable and a quadratic cone
sdo1.ptf	A simple semidefinite problem with one matrix variable and a quadratic cone
sensitivity.ssp	Sensitivity analysis specification for <code>transport.lp</code>
transport.lp	A linear problem in the sensitivity analysis example

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

Chapter 14

Interface changes

The section shows interface-specific changes to the **MOSEK** Command Line Tools in version 10.0 compared to version 9. See the [release notes](#) for general changes and new features of the **MOSEK** Optimization Suite.

14.1 Important changes compared to version 9

- **Parameters.** Users who set parameters to tune the performance and numerical properties of the solver (termination criteria, tolerances, solving primal or dual, presolve etc.) are recommended to reevaluate such tuning. It may be that other, or default, parameter settings will be more beneficial in the current version. The hints in [Sec. 7](#) may be useful for some cases.
- **Multithreading.** In the interior-point optimizer it is possible to set the number of threads with `MSK_IPAR_NUM_THREADS` before each optimization, and not just once per process. The parameter `MSK_IPAR_INTPNT_MULTI_THREAD` is no longer relevant and was removed.

14.2 Changes compared to version 9

14.3 Parameters compared to version 9

Added

- `MSK_DPAR_MIO_DJC_MAX_BIGM`
- `MSK_DPAR_PRESOLVE_TOL_PRIMAL_INFEAS_PERTURBATION`
- `MSK_IPAR_MIO_CONSTRUCT_SOL`
- `MSK_IPAR_MIO_CUT_LIPRO`
- `MSK_IPAR_MIO_DATA_PERMUTATION_METHOD`
- `MSK_IPAR_MIO_MEMORY_EMPHASIS_LEVEL`
- `MSK_IPAR_MIO_NUMERICAL_EMPHASIS_LEVEL`
- `MSK_IPAR_MIO_PRESOLVE_AGGREGATOR_USE`
- `MSK_IPAR_MIO_QCQO_REFORMULATION_METHOD`
- `MSK_IPAR_MIO_SYMMETRY_LEVEL`
- `MSK_IPAR_NG`
- `MSK_IPAR_PTF_WRITE_PARAMETERS`
- `MSK_IPAR_PTF_WRITE_SOLUTIONS`

- *MSK_IPAR_REMOTE_USE_COMPRESSION*
- *MSK_IPAR_SIM_DETECT_PWL*
- *MSK_IPAR_WRITE_JSON_INDENTATION*
- *MSK_SPAR_REMOTE_OPTSERVER_HOST*
- *MSK_SPAR_REMOTE_TLS_CERT*
- *MSK_SPAR_REMOTE_TLS_CERT_PATH*

Removed

- *MSK_IPAR_INTPNT_MULTI_THREAD*
- *MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU*
- *MSK_IPAR_READ_LP_QUOTED_NAMES*
- *MSK_IPAR_WRITE_LP_QUOTED_NAMES*
- *MSK_IPAR_WRITE_LP_STRICT_FORMAT*
- *MSK_IPAR_WRITE_LP_TERMS_PER_LINE*
- *MSK_IPAR_WRITE_PRECISION*
- *MSK_SPAR_REMOTE_ACCESS_TOKEN*
- *MSK_SPAR_STAT_FILE_NAME*

14.4 Constants compared to version 9

Added

- *MSK_CALLBACK_BEGIN_SOLVE_ROOT_RELAX*
- *MSK_CALLBACK_END_SOLVE_ROOT_RELAX*
- *MSK_CALLBACK_UPDATE_SIMPLEX*
- *MSK_DINF_ANA_PRO_SCALARIZED_CONSTRAINT_MATRIX_DENSITY*
- *MSK_DINF_MIO_INITIAL_FEASIBLE_SOLUTION_OBJ*
- *MSK_DINF_MIO_LIPRO_SEPARATION_TIME*
- *MSK_DINF_MIO_ROOT_TIME*
- *MSK_DINF_PRESOLVE_TOTAL_PRIMAL_PERTURBATION*
- *MSK_DINF_READ_DATA_TIME*
- *MSK_DINF_REMOTE_TIME*
- *MSK_DINF_SOL_ITG_PVIOLACC*
- *MSK_DINF_SOL_ITG_PVIOLDJC*
- *MSK_DINF_SOL_ITR_DVIOLACC*
- *MSK_DINF_SOL_ITR_PVIOLACC*
- *MSK_DINF_WRITE_DATA_TIME*

- *MSK_IINF_MIO_INITIAL_FEASIBLE_SOLUTION*
- *MSK_IINF_MIO_NUM_LIPRO_CUTS*
- *MSK_IINF_MIO_NUMDJC*
- *MSK_IINF_MIO_PRESOLVED_NUMDJC*
- *MSK_IINF_PRESOLVE_NUM_PRIMAL_PERTURBATIONS*
- *MSK_LIINF_ANA_PRO_SCALARIZED_CONSTRAINT_MATRIX_NUM_COLUMNS*
- *MSK_LIINF_ANA_PRO_SCALARIZED_CONSTRAINT_MATRIX_NUM_NZ*
- *MSK_LIINF_ANA_PRO_SCALARIZED_CONSTRAINT_MATRIX_NUM_ROWS*
- *MSK_LIINF_MIO_NUM_DUAL_ILLPOSED_CER*
- *MSK_LIINF_MIO_NUM_PRIM_ILLPOSED_CER*
- *MSK_LIINF_RD_NUMACC*
- *MSK_LIINF_RD_NUMDJC*
- *MSK_LIINF_SIMPLEX_ITER*

Removed

- *MSK_CALLBACKCODE_BEGIN_FULL_CONVEXITY_CHECK*
- *MSK_CALLBACKCODE_END_FULL_CONVEXITY_CHECK*
- *MSK_CALLBACKCODE_IM_FULL_CONVEXITY_CHECK*
- *MSK_DINFITEM_RD_TIME*
- *MSK_SCALINGTYPE_AGGRESSIVE*
- *MSK_SCALINGTYPE_MODERATE*

14.5 Response Codes compared to version 9

Added

- *MSK_RES_ERR_ACC_AFE_DOMAIN_MISMATCH*
- *MSK_RES_ERR_ACC_INVALID_ENTRY_INDEX*
- *MSK_RES_ERR_ACC_INVALID_INDEX*
- *MSK_RES_ERR_AFE_INVALID_INDEX*
- *MSK_RES_ERR_ARGUMENT_IS_TOO_SMALL*
- *MSK_RES_ERR_AXIS_NAME_SPECIFICATION*
- *MSK_RES_ERR_CBF_DUPLICATE_PSDCON*
- *MSK_RES_ERR_CBF_INVALID_DIMENSION_OF_PSDCON*
- *MSK_RES_ERR_CBF_INVALID_NUM_PSDCON*
- *MSK_RES_ERR_CBF_INVALID_PSDCON_BLOCK_INDEX*
- *MSK_RES_ERR_CBF_INVALID_PSDCON_INDEX*

- *MSK_RES_ERR_CBF_INVALID_PSDCON_VARIABLE_INDEX*
- *MSK_RES_ERR_CBF_UNSUPPORTED_CHANGE*
- *MSK_RES_ERR_DIMENSION_SPECIFICATION*
- *MSK_RES_ERR_DJC_AFE_DOMAIN_MISMATCH*
- *MSK_RES_ERR_DJC_DOMAIN_TERMSIZE_MISMATCH*
- *MSK_RES_ERR_DJC_INVALID_INDEX*
- *MSK_RES_ERR_DJC_INVALID_TERM_SIZE*
- *MSK_RES_ERR_DJC_TOTAL_NUM_TERMS_MISMATCH*
- *MSK_RES_ERR_DJC_UNSUPPORTED_DOMAIN_TYPE*
- *MSK_RES_ERR_DOMAIN_DIMENSION*
- *MSK_RES_ERR_DOMAIN_DIMENSION_PSD*
- *MSK_RES_ERR_DOMAIN_INVALID_INDEX*
- *MSK_RES_ERR_DOMAIN_POWER_INVALID_ALPHA*
- *MSK_RES_ERR_DOMAIN_POWER_NEGATIVE_ALPHA*
- *MSK_RES_ERR_DOMAIN_POWER_NLEFT*
- *MSK_RES_ERR_DUPLICATE_DJC_NAMES*
- *MSK_RES_ERR_DUPLICATE_DOMAIN_NAMES*
- *MSK_RES_ERR_DUPLICATE_FIJ*
- *MSK_RES_ERR_HUGE_FIJ*
- *MSK_RES_ERR_INDEX_IS_NOT_UNIQUE*
- *MSK_RES_ERR_INF_IN_DOUBLE_DATA*
- *MSK_RES_ERR_INVALID_B*
- *MSK_RES_ERR_INVALID_CFIX*
- *MSK_RES_ERR_INVALID_FIJ*
- *MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_AFFINE_CONIC_CONSTRAINTS*
- *MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_DISJUNCTIVE_CONSTRAINTS*
- *MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_QUADRATIC_TERMS*
- *MSK_RES_ERR_INVALID_G*
- *MSK_RES_ERR_LICENSE_OLD_SERVER_VERSION*
- *MSK_RES_ERR_LP_INDICATOR_VAR*
- *MSK_RES_ERR_MISMATCHING_DIMENSION*
- *MSK_RES_ERR_MPS_INVALID_BOUND_KEY*
- *MSK_RES_ERR_MPS_INVALID_CON_KEY*
- *MSK_RES_ERR_MPS_INVALID_INDICATOR_CONSTRAINT*
- *MSK_RES_ERR_MPS_INVALID_INDICATOR_QUADRATIC_CONSTRAINT*
- *MSK_RES_ERR_MPS_INVALID_INDICATOR_VALUE*

- *MSK_RES_ERR_MPS_INVALID_INDICATOR_VARIABLE*
- *MSK_RES_ERR_MPS_INVALID_KEY*
- *MSK_RES_ERR_MPS_INVALID_SEC_NAME*
- *MSK_RES_ERR_MPS_WRITE_CPLEX_INVALID_CONE_TYPE*
- *MSK_RES_ERR_NO_DOTY*
- *MSK_RES_ERR_NOT_POWER_DOMAIN*
- *MSK_RES_ERR_OPF_DUAL_INTEGER_SOLUTION*
- *MSK_RES_ERR_OPF_DUPLICATE_BOUND*
- *MSK_RES_ERR_OPF_DUPLICATE_CONE_ENTRY*
- *MSK_RES_ERR_OPF_DUPLICATE_CONSTRAINT_NAME*
- *MSK_RES_ERR_OPF_INCORRECT_TAG_PARAM*
- *MSK_RES_ERR_OPF_INVALID_CONE_TYPE*
- *MSK_RES_ERR_OPF_INVALID_TAG*
- *MSK_RES_ERR_OPF_MISMATCHED_TAG*
- *MSK_RES_ERR_OPF_SYNTAX*
- *MSK_RES_ERR_OPF_TOO_LARGE*
- *MSK_RES_ERR_PTF_INCOMPATIBILITY*
- *MSK_RES_ERR_PTF_INCONSISTENCY*
- *MSK_RES_ERR_PTF_UNDEFINED_ITEM*
- *MSK_RES_ERR_SERVER_ACCESS_TOKEN*
- *MSK_RES_ERR_SERVER_ADDRESS*
- *MSK_RES_ERR_SERVER_CERTIFICATE*
- *MSK_RES_ERR_SERVER_TLS_CLIENT*
- *MSK_RES_ERR_SPARSITY_SPECIFICATION*
- *MSK_RES_ERR_UNALLOWED_WHICH SOL*
- *MSK_RES_TRM_LOST_RACE*
- *MSK_RES_WRN_INVALID_MPS_NAME*
- *MSK_RES_WRN_INVALID_MPS_OBJ_NAME*
- *MSK_RES_WRN_LARGE_FIJ*
- *MSK_RES_WRN_MODIFIED_DOUBLE_PARAMETER*
- *MSK_RES_WRN_NO_INFEASIBILITY_REPORT_WHEN_MATRIX_VARIABLES*
- *MSK_RES_WRN_PRESOLVE_PRIMAL_PERTUBATIONS*
- *MSK_RES_WRN_WRITE_LP_DUPLICATE_CON_NAMES*
- *MSK_RES_WRN_WRITE_LP_DUPLICATE_VAR_NAMES*
- *MSK_RES_WRN_WRITE_LP_INVALID_CON_NAMES*
- *MSK_RES_WRN_WRITE_LP_INVALID_VAR_NAMES*

Removed

- MSK_RES_ERR_LP_FORMAT
- MSK_RES_ERR_MPS_INV_BOUND_KEY
- MSK_RES_ERR_MPS_INV_CON_KEY
- MSK_RES_ERR_MPS_INV_SEC_NAME
- MSK_RES_ERR_OPF_FORMAT
- MSK_RES_ERR_OPF_NEW_VARIABLE
- MSK_RES_WRN_EXP_CONES_WITH_VARIABLES_FIXED_AT_ZERO
- MSK_RES_WRN_POW_CONES_WITH_ROOT_FIXED_AT_ZERO
- MSK_RES_WRN_QUAD_CONES_WITH_ROOT_FIXED_AT_ZERO
- MSK_RES_WRN_RQUAD_CONES_WITH_ROOT_FIXED_AT_ZERO

Bibliography

- [AA95] E. D. Andersen and K. D. Andersen. Presolving in linear programming. *Math. Programming*, 71(2):221–245, 1995.
- [AGMeszarosX96] E. D. Andersen, J. Gondzio, Cs. Mészáros, and X. Xu. Implementation of interior point methods for large scale linear programming. In T. Terlaky, editor, *Interior-point methods of mathematical programming*, pages 189–252. Kluwer Academic Publishers, 1996.
- [ART03] E. D. Andersen, C. Roos, and T. Terlaky. On implementing a primal-dual interior-point method for conic quadratic optimization. *Math. Programming*, February 2003.
- [AY96] E. D. Andersen and Y. Ye. Combining interior-point and pivoting algorithms. *Management Sci.*, 42(12):1719–1731, December 1996.
- [And09] Erling D. Andersen. The homogeneous and self-dual model and algorithm for linear optimization. Technical Report TR-1-2009, MOSEK ApS, 2009. URL: <http://docs.mosek.com/whitepapers/homolo.pdf>.
- [And13] Erling D. Andersen. On formulating quadratic functions in optimization models. Technical Report TR-1-2013, MOSEK ApS, 2013. Last revised 23-feb-2016. URL: <http://docs.mosek.com/whitepapers/qmodel.pdf>.
- [Chvatal83] V. Chvátal. *Linear programming*. W.H. Freeman and Company, 1983.
- [CCornuejolsZ14] M. Conforti, G. Cornuéjols, and G. Zambelli. *Integer programming*. Springer, 2014.
- [FGK03] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL. A modeling language for mathematical programming*. Thomson, 2nd edition, 2003.
- [Naz87] J. L. Nazareth. *Computer Solution of Linear Programs*. Oxford University Press, New York, 1987.
- [RTV97] C. Roos, T. Terlaky, and J. -Ph. Vial. *Theory and algorithms for linear optimization: an interior point approach*. John Wiley and Sons, New York, 1997.
- [Wal00] S. W. Wallace. Decision making under uncertainty: is sensitivity of any use. *Oper. Res.*, 48(1):20–25, January 2000.
- [Wol98] L. A. Wolsey. *Integer programming*. John Wiley and Sons, 1998.

Symbol Index

Functions

Parameters

Double parameters, 101

MSK_DPAR_ANA_SOL_INFEAS_TOL, 101
MSK_DPAR_BASIS_REL_TOL_S, 101
MSK_DPAR_BASIS_TOL_S, 101
MSK_DPAR_BASIS_TOL_X, 101
MSK_DPAR_CHECK_CONVEXITY_REL_TOL, 101
MSK_DPAR_DATA_SYM_MAT_TOL, 102
MSK_DPAR_DATA_SYM_MAT_TOL_HUGE, 102
MSK_DPAR_DATA_SYM_MAT_TOL_LARGE, 102
MSK_DPAR_DATA_TOL_AIJ_HUGE, 102
MSK_DPAR_DATA_TOL_AIJ_LARGE, 102
MSK_DPAR_DATA_TOL_BOUND_INF, 102
MSK_DPAR_DATA_TOL_BOUND_WRN, 102
MSK_DPAR_DATA_TOL_C_HUGE, 103
MSK_DPAR_DATA_TOL_CJ_LARGE, 103
MSK_DPAR_DATA_TOL_QIJ, 103
MSK_DPAR_DATA_TOL_X, 103
MSK_DPAR_INTPNT_CO_TOL_DFEAS, 103
MSK_DPAR_INTPNT_CO_TOL_INFEAS, 103
MSK_DPAR_INTPNT_CO_TOL_MU_RED, 103
MSK_DPAR_INTPNT_CO_TOL_NEAR_REL, 104
MSK_DPAR_INTPNT_CO_TOL_PFEAS, 104
MSK_DPAR_INTPNT_CO_TOL_REL_GAP, 104
MSK_DPAR_INTPNT_QO_TOL_DFEAS, 104
MSK_DPAR_INTPNT_QO_TOL_INFEAS, 104
MSK_DPAR_INTPNT_QO_TOL_MU_RED, 104
MSK_DPAR_INTPNT_QO_TOL_NEAR_REL, 105
MSK_DPAR_INTPNT_QO_TOL_PFEAS, 105
MSK_DPAR_INTPNT_QO_TOL_REL_GAP, 105
MSK_DPAR_INTPNT_TOL_DFEAS, 105
MSK_DPAR_INTPNT_TOL_DSAFE, 105
MSK_DPAR_INTPNT_TOL_INFEAS, 105
MSK_DPAR_INTPNT_TOL_MU_RED, 106
MSK_DPAR_INTPNT_TOL_PATH, 106
MSK_DPAR_INTPNT_TOL_PFEAS, 106
MSK_DPAR_INTPNT_TOL_PSAFE, 106
MSK_DPAR_INTPNT_TOL_REL_GAP, 106
MSK_DPAR_INTPNT_TOL_REL_STEP, 106
MSK_DPAR_INTPNT_TOL_STEP_SIZE, 106
MSK_DPAR_LOWER_OBJ_CUT, 107
MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH, 107
MSK_DPAR_MIO_DJC_MAX_BIGM, 107
MSK_DPAR_MIO_MAX_TIME, 107
MSK_DPAR_MIO_REL_GAP_CONST, 107
MSK_DPAR_MIO_TOL_ABS_GAP, 107
MSK_DPAR_MIO_TOL_ABS_RELAX_INT, 107

MSK_DPAR_MIO_TOL_FEAS, 108
MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT,
108
MSK_DPAR_MIO_TOL_REL_GAP, 108
MSK_DPAR_OPTIMIZER_MAX_TIME, 108
MSK_DPAR_PREOLVE_TOL_ABS_LINDEP, 108
MSK_DPAR_PREOLVE_TOL_AIJ, 108
MSK_DPAR_PREOLVE_TOL_PRIMAL_INFEAS_PERTURBATION,
108
MSK_DPAR_PREOLVE_TOL_REL_LINDEP, 109
MSK_DPAR_PREOLVE_TOL_S, 109
MSK_DPAR_PREOLVE_TOL_X, 109
MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL, 109
MSK_DPAR_SEMIDEFINITE_TOL_APPROX, 109
MSK_DPAR_SIM_LU_TOL_REL_PIV, 109
MSK_DPAR_SIMPLEX_ABS_TOL_PIV, 110
MSK_DPAR_UPPER_OBJ_CUT, 110
MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH, 110
Integer parameters, 110
MSK_IPAR_ANA_SOL_BASIS, 110
MSK_IPAR_ANA_SOL_PRINT_VIOLATED, 110
MSK_IPAR_AUTO_SORT_A_BEFORE_OPT, 110
MSK_IPAR_AUTO_UPDATE_SOL_INFO, 110
MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE, 111
MSK_IPAR_BI_CLEAN_OPTIMIZER, 111
MSK_IPAR_BI_IGNORE_MAX_ITER, 111
MSK_IPAR_BI_IGNORE_NUM_ERROR, 111
MSK_IPAR_BI_MAX_ITERATIONS, 111
MSK_IPAR_CACHE_LICENSE, 111
MSK_IPAR_CHECK_CONVEXITY, 112
MSK_IPAR_COMPRESS_STATFILE, 112
MSK_IPAR_INFEAS_GENERIC_NAMES, 112
MSK_IPAR_INFEAS_PREFER_PRIMAL, 112
MSK_IPAR_INFEAS_REPORT_AUTO, 112
MSK_IPAR_INFEAS_REPORT_LEVEL, 112
MSK_IPAR_INTPNT_BASIS, 112
MSK_IPAR_INTPNT_DIFF_STEP, 113
MSK_IPAR_INTPNT_HOTSTART, 113
MSK_IPAR_INTPNT_MAX_ITERATIONS, 113
MSK_IPAR_INTPNT_MAX_NUM_COR, 113
MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS,
113
MSK_IPAR_INTPNT_OFF_COL_TRH, 113
MSK_IPAR_INTPNT_ORDER_GP_NUM_SEEDS, 114
MSK_IPAR_INTPNT_ORDER_METHOD, 114
MSK_IPAR_INTPNT_PURIFY, 114
MSK_IPAR_INTPNT_REGULARIZATION_USE, 114
MSK_IPAR_INTPNT_SCALING, 114
MSK_IPAR_INTPNT_SOLVE_FORM, 114

MSK_IPAR_INTPNT_STARTING_POINT, 114
 MSK_IPAR_LICENSE_DEBUG, 115
 MSK_IPAR_LICENSE_PAUSE_TIME, 115
 MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS, 115
 MSK_IPAR_LICENSE_TRH_EXPIRY_WRN, 115
 MSK_IPAR_LICENSE_WAIT, 115
 MSK_IPAR_LOG, 115
 MSK_IPAR_LOG_ANA_PRO, 116
 MSK_IPAR_LOG_BI, 116
 MSK_IPAR_LOG_BI_FREQ, 116
 MSK_IPAR_LOG_CHECK_CONVEXITY, 116
 MSK_IPAR_LOG_CUT_SECOND_OPT, 116
 MSK_IPAR_LOG_EXPAND, 116
 MSK_IPAR_LOG_FEAS_REPAIR, 116
 MSK_IPAR_LOG_FILE, 117
 MSK_IPAR_LOG_INCLUDE_SUMMARY, 117
 MSK_IPAR_LOG_INFEAS_ANA, 117
 MSK_IPAR_LOG_INTPNT, 117
 MSK_IPAR_LOG_LOCAL_INFO, 117
 MSK_IPAR_LOG_MIO, 117
 MSK_IPAR_LOG_MIO_FREQ, 118
 MSK_IPAR_LOG_ORDER, 118
 MSK_IPAR_LOG_PREOLVE, 118
 MSK_IPAR_LOG_RESPONSE, 118
 MSK_IPAR_LOG_SENSITIVITY, 118
 MSK_IPAR_LOG_SENSITIVITY_OPT, 118
 MSK_IPAR_LOG_SIM, 118
 MSK_IPAR_LOG_SIM_FREQ, 119
 MSK_IPAR_LOG_SIM_MINOR, 119
 MSK_IPAR_LOG_STORAGE, 119
 MSK_IPAR_MAX_NUM_WARNINGS, 119
 MSK_IPAR_MIO_BRANCH_DIR, 119
 MSK_IPAR_MIO_CONIC_OUTER_APPROXIMATION, 119
 MSK_IPAR_MIO_CONSTRUCT_SOL, 119
 MSK_IPAR_MIO_CUT_CLIQUE, 120
 MSK_IPAR_MIO_CUT_CMIR, 120
 MSK_IPAR_MIO_CUT_GMI, 120
 MSK_IPAR_MIO_CUT_IMPLIED_BOUND, 120
 MSK_IPAR_MIO_CUT_KNAPSACK_COVER, 120
 MSK_IPAR_MIO_CUT_LIPRO, 120
 MSK_IPAR_MIO_CUT_SELECTION_LEVEL, 120
 MSK_IPAR_MIO_DATA_PERMUTATION_METHOD, 121
 MSK_IPAR_MIO_FEASPUMP_LEVEL, 121
 MSK_IPAR_MIO_HEURISTIC_LEVEL, 121
 MSK_IPAR_MIO_MAX_NUM_BRANCHES, 121
 MSK_IPAR_MIO_MAX_NUM_RELAXS, 121
 MSK_IPAR_MIO_MAX_NUM_ROOT_CUT_ROUNDS, 122
 MSK_IPAR_MIO_MAX_NUM_SOLUTIONS, 122
 MSK_IPAR_MIO_MEMORY_EMPHASIS_LEVEL, 122
 MSK_IPAR_MIO_MODE, 122
 MSK_IPAR_MIO_NODE_OPTIMIZER, 122
 MSK_IPAR_MIO_NODE_SELECTION, 122
 MSK_IPAR_MIO_NUMERICAL_EMPHASIS_LEVEL, 122
 MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE, 123
 MSK_IPAR_MIO_PREOLVE_AGGREGATOR_USE, 123
 MSK_IPAR_MIO_PROBING_LEVEL, 123
 MSK_IPAR_MIO_PROPAGATE_OBJECTIVE_CONSTRAINT, 123
 MSK_IPAR_MIO_QCQO_REFORMULATION_METHOD, 123
 MSK_IPAR_MIO_RINS_MAX_NODES, 124
 MSK_IPAR_MIO_ROOT_OPTIMIZER, 124
 MSK_IPAR_MIO_ROOT_REPEAT_PREOLVE_LEVEL, 124
 MSK_IPAR_MIO_SEED, 124
 MSK_IPAR_MIO_SYMMETRY_LEVEL, 124
 MSK_IPAR_MIO_VB_DETECTION_LEVEL, 124
 MSK_IPAR_MT_SPINCOUNT, 125
 MSK_IPAR_NG, 125
 MSK_IPAR_NUM_THREADS, 125
 MSK_IPAR_OPF_WRITE_HEADER, 125
 MSK_IPAR_OPF_WRITE_HINTS, 125
 MSK_IPAR_OPF_WRITE_LINE_LENGTH, 125
 MSK_IPAR_OPF_WRITE_PARAMETERS, 126
 MSK_IPAR_OPF_WRITE_PROBLEM, 126
 MSK_IPAR_OPF_WRITE_SOL_BAS, 126
 MSK_IPAR_OPF_WRITE_SOL_ITG, 126
 MSK_IPAR_OPF_WRITE_SOL_ITR, 126
 MSK_IPAR_OPF_WRITE_SOLUTIONS, 126
 MSK_IPAR_OPTIMIZER, 126
 MSK_IPAR_PARAM_READ_CASE_NAME, 127
 MSK_IPAR_PARAM_READ_IGN_ERROR, 127
 MSK_IPAR_PREOLVE_ELIMINATOR_MAX_FILL, 127
 MSK_IPAR_PREOLVE_ELIMINATOR_MAX_NUM_TRIES, 127
 MSK_IPAR_PREOLVE_LEVEL, 127
 MSK_IPAR_PREOLVE_LINDEP_ABS_WORK_TRH, 127
 MSK_IPAR_PREOLVE_LINDEP_REL_WORK_TRH, 127
 MSK_IPAR_PREOLVE_LINDEP_USE, 128
 MSK_IPAR_PREOLVE_MAX_NUM_PASS, 128
 MSK_IPAR_PREOLVE_MAX_NUM_REDUCTIONS, 128
 MSK_IPAR_PREOLVE_USE, 128
 MSK_IPAR_PRIMAL_REPAIR_OPTIMIZER, 128
 MSK_IPAR_PTF_WRITE_PARAMETERS, 128
 MSK_IPAR_PTF_WRITE_SOLUTIONS, 128
 MSK_IPAR_PTF_WRITE_TRANSFORM, 129
 MSK_IPAR_READ_DEBUG, 129
 MSK_IPAR_READ_KEEP_FREE_CON, 129
 MSK_IPAR_READ_MPS_FORMAT, 129
 MSK_IPAR_READ_MPS_WIDTH, 129
 MSK_IPAR_READ_TASK_IGNORE_PARAM, 129
 MSK_IPAR_REMOTE_USE_COMPRESSION, 129
 MSK_IPAR_REMOVE_UNUSED_SOLUTIONS, 130
 MSK_IPAR_SENSITIVITY_ALL, 130
 MSK_IPAR_SENSITIVITY_OPTIMIZER, 130
 MSK_IPAR_SENSITIVITY_TYPE, 130
 MSK_IPAR_SIM_BASIS_FACTOR_USE, 130
 MSK_IPAR_SIM_DEGEN, 130
 MSK_IPAR_SIM_DETECT_PWL, 130
 MSK_IPAR_SIM_DUAL_CRASH, 131
 MSK_IPAR_SIM_DUAL_PHASEONE_METHOD, 131
 MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION, 131
 MSK_IPAR_SIM_DUAL_SELECTION, 131
 MSK_IPAR_SIM_EXPLOIT_DUPVEC, 131
 MSK_IPAR_SIM_HOTSTART, 131
 MSK_IPAR_SIM_HOTSTART_LU, 132
 MSK_IPAR_SIM_MAX_ITERATIONS, 132

MSK_IPAR_SIM_MAX_NUM_SETBACKS, 132
 MSK_IPAR_SIM_NON_SINGULAR, 132
 MSK_IPAR_SIM_PRIMAL_CRASH, 132
 MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD, 132
 MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION, 132
 MSK_IPAR_SIM_PRIMAL_SELECTION, 133
 MSK_IPAR_SIM_REFACTOR_FREQ, 133
 MSK_IPAR_SIM_REFORMULATION, 133
 MSK_IPAR_SIM_SAVE_LU, 133
 MSK_IPAR_SIM_SCALING, 133
 MSK_IPAR_SIM_SCALING_METHOD, 133
 MSK_IPAR_SIM_SEED, 134
 MSK_IPAR_SIM_SOLVE_FORM, 134
 MSK_IPAR_SIM_STABILITY_PRIORITY, 134
 MSK_IPAR_SIM_SWITCH_OPTIMIZER, 134
 MSK_IPAR_SOL_FILTER_KEEP_BASIC, 134
 MSK_IPAR_SOL_FILTER_KEEP_RANGED, 134
 MSK_IPAR_SOL_READ_NAME_WIDTH, 135
 MSK_IPAR_SOL_READ_WIDTH, 135
 MSK_IPAR_SOLUTION_CALLBACK, 135
 MSK_IPAR_TIMING_LEVEL, 135
 MSK_IPAR_WRITE_BAS_CONSTRAINTS, 135
 MSK_IPAR_WRITE_BAS_HEAD, 135
 MSK_IPAR_WRITE_BAS_VARIABLES, 135
 MSK_IPAR_WRITE_COMPRESSION, 135
 MSK_IPAR_WRITE_DATA_PARAM, 136
 MSK_IPAR_WRITE_FREE_CON, 136
 MSK_IPAR_WRITE_GENERIC_NAMES, 136
 MSK_IPAR_WRITE_GENERIC_NAMES_IO, 136
 MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_ITEMS,
 136
 MSK_IPAR_WRITE_INT_CONSTRAINTS, 136
 MSK_IPAR_WRITE_INT_HEAD, 137
 MSK_IPAR_WRITE_INT_VARIABLES, 137
 MSK_IPAR_WRITE_JSON_INDENTATION, 137
 MSK_IPAR_WRITE_LP_FULL_OBJ, 137
 MSK_IPAR_WRITE_LP_LINE_WIDTH, 137
 MSK_IPAR_WRITE_MPS_FORMAT, 137
 MSK_IPAR_WRITE_MPS_INT, 137
 MSK_IPAR_WRITE_SOL_BARVARIABLES, 137
 MSK_IPAR_WRITE_SOL_CONSTRAINTS, 138
 MSK_IPAR_WRITE_SOL_HEAD, 138
 MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES,
 138
 MSK_IPAR_WRITE_SOL_VARIABLES, 138
 MSK_IPAR_WRITE_TASK_INC_SOL, 138
 MSK_IPAR_WRITE_XML_MODE, 138
 String parameters, 139
 MSK_SPAR_BAS_SOL_FILE_NAME, 139
 MSK_SPAR_DATA_FILE_NAME, 139
 MSK_SPAR_DEBUG_FILE_NAME, 139
 MSK_SPAR_INT_SOL_FILE_NAME, 139
 MSK_SPAR_ITR_SOL_FILE_NAME, 139
 MSK_SPAR_MIO_DEBUG_STRING, 139
 MSK_SPAR_PARAM_COMMENT_SIGN, 139
 MSK_SPAR_PARAM_READ_FILE_NAME, 139
 MSK_SPAR_PARAM_WRITE_FILE_NAME, 139
 MSK_SPAR_READ_MPS_BOU_NAME, 140

MSK_SPAR_READ_MPS_OBJ_NAME, 140
 MSK_SPAR_READ_MPS_RAN_NAME, 140
 MSK_SPAR_READ_MPS_RHS_NAME, 140
 MSK_SPAR_REMOTE_OPTSERVER_HOST, 140
 MSK_SPAR_REMOTE_TLS_CERT, 140
 MSK_SPAR_REMOTE_TLS_CERT_PATH, 140
 MSK_SPAR_SENSITIVITY_FILE_NAME, 140
 MSK_SPAR_SENSITIVITY_RES_FILE_NAME, 141
 MSK_SPAR_SOL_FILTER_XC_LOW, 141
 MSK_SPAR_SOL_FILTER_XC_UPR, 141
 MSK_SPAR_SOL_FILTER_XX_LOW, 141
 MSK_SPAR_SOL_FILTER_XX_UPR, 141
 MSK_SPAR_STAT_KEY, 141
 MSK_SPAR_STAT_NAME, 141
 MSK_SPAR_WRITE_LP_GEN_VAR_NAME, 142

Response codes

Termination, 142
 MSK_RES_OK, 142
 MSK_RES_TRM_INTERNAL, 143
 MSK_RES_TRM_INTERNAL_STOP, 143
 MSK_RES_TRM_LOST_RACE, 143
 MSK_RES_TRM_MAX_ITERATIONS, 142
 MSK_RES_TRM_MAX_NUM_SETBACKS, 142
 MSK_RES_TRM_MAX_TIME, 142
 MSK_RES_TRM_MIO_NUM_BRANCHES, 142
 MSK_RES_TRM_MIO_NUM_RELAXS, 142
 MSK_RES_TRM_NUM_MAX_NUM_INT_SOLUTIONS, 142
 MSK_RES_TRM_NUMERICAL_PROBLEM, 142
 MSK_RES_TRM_OBJECTIVE_RANGE, 142
 MSK_RES_TRM_STALL, 142
 MSK_RES_TRM_USER_CALLBACK, 142
 Warnings, 143
 MSK_RES_WRN_ANA_ALMOST_INT_BOUNDS, 145
 MSK_RES_WRN_ANA_C_ZERO, 145
 MSK_RES_WRN_ANA_CLOSE_BOUNDS, 145
 MSK_RES_WRN_ANA_EMPTY_COLS, 145
 MSK_RES_WRN_ANA_LARGE_BOUNDS, 145
 MSK_RES_WRN_DROPPED_NZ_QOBJ, 143
 MSK_RES_WRN_DUPLICATE_BARVARIABLE_NAMES,
 144
 MSK_RES_WRN_DUPLICATE_CONE_NAMES, 145
 MSK_RES_WRN_DUPLICATE_CONSTRAINT_NAMES, 144
 MSK_RES_WRN_DUPLICATE_VARIABLE_NAMES, 144
 MSK_RES_WRN_ELIMINATOR_SPACE, 144
 MSK_RES_WRN_EMPTY_NAME, 144
 MSK_RES_WRN_IGNORE_INTEGER, 143
 MSK_RES_WRN_INCOMPLETE_LINEAR_DEPENDENCY_CHECK,
 144
 MSK_RES_WRN_INVALID_MPS_NAME, 144
 MSK_RES_WRN_INVALID_MPS_OBJ_NAME, 144
 MSK_RES_WRN_LARGE_AIJ, 143
 MSK_RES_WRN_LARGE_BOUND, 143
 MSK_RES_WRN_LARGE_CJ, 143
 MSK_RES_WRN_LARGE_CON_FX, 143
 MSK_RES_WRN_LARGE_FIJ, 145
 MSK_RES_WRN_LARGE_LO_BOUND, 143
 MSK_RES_WRN_LARGE_UP_BOUND, 143

MSK_RES_WRN_LICENSE_EXPIRE, 144
 MSK_RES_WRN_LICENSE_FEATURE_EXPIRE, 144
 MSK_RES_WRN_LICENSE_SERVER, 144
 MSK_RES_WRN_LP_DROP_VARIABLE, 143
 MSK_RES_WRN_LP_OLD_QUAD_FORMAT, 143
 MSK_RES_WRN_MIO_INFEASIBLE_FINAL, 143
 MSK_RES_WRN_MODIFIED_DOUBLE_PARAMETER, 145
 MSK_RES_WRN_MPS_SPLIT_BOU_VECTOR, 143
 MSK_RES_WRN_MPS_SPLIT_RAN_VECTOR, 143
 MSK_RES_WRN_MPS_SPLIT_RHS_VECTOR, 143
 MSK_RES_WRN_NAME_MAX_LEN, 143
 MSK_RES_WRN_NO_DUALIZER, 145
 MSK_RES_WRN_NO_GLOBAL_OPTIMIZER, 143
 MSK_RES_WRN_NO_INFEASIBILITY_REPORT_WHEN_MATRIBES, 145
 MSK_RES_WRN_NZ_IN_UPR_TRI, 143
 MSK_RES_WRN_OPEN_PARAM_FILE, 143
 MSK_RES_WRN_PARAM_IGNORED_CMIO, 144
 MSK_RES_WRN_PARAM_NAME_DOU, 144
 MSK_RES_WRN_PARAM_NAME_INT, 144
 MSK_RES_WRN_PARAM_NAME_STR, 144
 MSK_RES_WRN_PARAM_STR_VALUE, 144
 MSK_RES_WRN_PRESOLVE_OUTOFSPACE, 144
 MSK_RES_WRN_PRESOLVE_PRIMAL_PERTUBATIONS, 144
 MSK_RES_WRN_SOL_FILE_IGNORED_CON, 143
 MSK_RES_WRN_SOL_FILE_IGNORED_VAR, 144
 MSK_RES_WRN_SOL_FILTER, 143
 MSK_RES_WRN_SPAR_MAX_LEN, 143
 MSK_RES_WRN_SYM_MAT_LARGE, 145
 MSK_RES_WRN_TOO_FEW_BASIS_VARS, 144
 MSK_RES_WRN_TOO_MANY_BASIS_VARS, 144
 MSK_RES_WRN_UNDEF_SOL_FILE_NAME, 143
 MSK_RES_WRN_USING_GENERIC_NAMES, 144
 MSK_RES_WRN_WRITE_CHANGED_NAMES, 144
 MSK_RES_WRN_WRITE_DISCARDED_CFIX, 144
 MSK_RES_WRN_WRITE_LP_DUPLICATE_CON_NAMES, 145
 MSK_RES_WRN_WRITE_LP_DUPLICATE_VAR_NAMES, 145
 MSK_RES_WRN_WRITE_LP_INVALID_CON_NAMES, 145
 MSK_RES_WRN_WRITE_LP_INVALID_VAR_NAMES, 145
 MSK_RES_WRN_ZERO_AIJ, 143
 MSK_RES_WRN_ZEROS_IN_SPARSE_COL, 144
 MSK_RES_WRN_ZEROS_IN_SPARSE_ROW, 144
 Errors, 145
 MSK_RES_ERR_ACC_AFE_DOMAIN_MISMATCH, 162
 MSK_RES_ERR_ACC_INVALID_ENTRY_INDEX, 162
 MSK_RES_ERR_ACC_INVALID_INDEX, 162
 MSK_RES_ERR_AD_INVALID_CODELIST, 157
 MSK_RES_ERR_AFE_INVALID_INDEX, 161
 MSK_RES_ERR_API_ARRAY_TOO_SMALL, 156
 MSK_RES_ERR_API_CB_CONNECT, 156
 MSK_RES_ERR_API_FATAL_ERROR, 156
 MSK_RES_ERR_API_INTERNAL, 157
 MSK_RES_ERR_APPENDING_TOO_BIG_CONE, 154
 MSK_RES_ERR_ARG_IS_TOO_LARGE, 151
 MSK_RES_ERR_ARG_IS_TOO_SMALL, 151
 MSK_RES_ERR_ARGUMENT_DIMENSION, 150
 MSK_RES_ERR_ARGUMENT_IS_TOO_LARGE, 158
 MSK_RES_ERR_ARGUMENT_IS_TOO_SMALL, 158
 MSK_RES_ERR_ARGUMENT_LENNEQ, 150
 MSK_RES_ERR_ARGUMENT_PERM_ARRAY, 153
 MSK_RES_ERR_ARGUMENT_TYPE, 150
 MSK_RES_ERR_AXIS_NAME_SPECIFICATION, 148
 MSK_RES_ERR_BAR_VAR_DIM, 157
 MSK_RES_ERR_BASIS, 152
 MSK_RES_ERR_BASIS_FACTOR, 155
 MSK_RES_ERR_BASIS_SINGULAR, 155
 MSK_RES_ERR_BLANK_NAME, 147
 MSK_RES_ERR_CBF_DUPLICATE_ACOORD, 159
 MSK_RES_ERR_CBF_DUPLICATE_BCOORD, 159
 MSK_RES_ERR_CBF_DUPLICATE_CON, 159
 MSK_RES_ERR_CBF_DUPLICATE_INT, 159
 MSK_RES_ERR_CBF_DUPLICATE_OBJ, 159
 MSK_RES_ERR_CBF_DUPLICATE_OBJACCOORD, 159
 MSK_RES_ERR_CBF_DUPLICATE_POW_CONES, 160
 MSK_RES_ERR_CBF_DUPLICATE_POW_STAR_CONES, 160
 MSK_RES_ERR_CBF_DUPLICATE_PSDCON, 160
 MSK_RES_ERR_CBF_DUPLICATE_PSDVAR, 160
 MSK_RES_ERR_CBF_DUPLICATE_VAR, 159
 MSK_RES_ERR_CBF_INVALID_CON_TYPE, 159
 MSK_RES_ERR_CBF_INVALID_DIMENSION_OF_CONES, 160
 MSK_RES_ERR_CBF_INVALID_DIMENSION_OF_PSDCON, 160
 MSK_RES_ERR_CBF_INVALID_DOMAIN_DIMENSION, 159
 MSK_RES_ERR_CBF_INVALID_EXP_DIMENSION, 160
 MSK_RES_ERR_CBF_INVALID_INT_INDEX, 160
 MSK_RES_ERR_CBF_INVALID_NUM_PSDCON, 160
 MSK_RES_ERR_CBF_INVALID_NUMBER_OF_CONES, 160
 MSK_RES_ERR_CBF_INVALID_POWER, 160
 MSK_RES_ERR_CBF_INVALID_POWER_CONE_INDEX, 160
 MSK_RES_ERR_CBF_INVALID_POWER_STAR_CONE_INDEX, 160
 MSK_RES_ERR_CBF_INVALID_PSDCON_BLOCK_INDEX, 160
 MSK_RES_ERR_CBF_INVALID_PSDCON_INDEX, 160
 MSK_RES_ERR_CBF_INVALID_PSDCON_VARIABLE_INDEX, 160
 MSK_RES_ERR_CBF_INVALID_PSDVAR_DIMENSION, 160
 MSK_RES_ERR_CBF_INVALID_VAR_TYPE, 159
 MSK_RES_ERR_CBF_NO_VARIABLES, 159
 MSK_RES_ERR_CBF_NO_VERSION_SPECIFIED, 159
 MSK_RES_ERR_CBF_OBJ_SENSE, 159
 MSK_RES_ERR_CBF_PARSE, 159
 MSK_RES_ERR_CBF_POWER_CONE_IS_TOO_LONG, 160
 MSK_RES_ERR_CBF_POWER_CONE_MISMATCH, 160
 MSK_RES_ERR_CBF_POWER_STAR_CONE_MISMATCH, 160
 MSK_RES_ERR_CBF_SYNTAX, 159

MSK_RES_ERR_CBF_TOO_FEW_CONSTRAINTS, 159
 MSK_RES_ERR_CBF_TOO_FEW_INTS, 159
 MSK_RES_ERR_CBF_TOO_FEW_PSDVAR, 160
 MSK_RES_ERR_CBF_TOO_FEW_VARIABLES, 159
 MSK_RES_ERR_CBF_TOO_MANY_CONSTRAINTS, 159
 MSK_RES_ERR_CBF_TOO_MANY_INTS, 160
 MSK_RES_ERR_CBF_TOO_MANY_VARIABLES, 159
 MSK_RES_ERR_CBF_UNHANDLED_POWER_CONE_TYPE, 160
 MSK_RES_ERR_CBF_UNHANDLED_POWER_STAR_CONE_TYPE, 160
 MSK_RES_ERR_CBF_UNSUPPORTED, 160
 MSK_RES_ERR_CBF_UNSUPPORTED_CHANGE, 160
 MSK_RES_ERR_CON_Q_NOT_NSD, 153
 MSK_RES_ERR_CON_Q_NOT_PSD, 153
 MSK_RES_ERR_CONE_INDEX, 153
 MSK_RES_ERR_CONE_OVERLAP, 153
 MSK_RES_ERR_CONE_OVERLAP_APPEND, 153
 MSK_RES_ERR_CONE_PARAMETER, 154
 MSK_RES_ERR_CONE_REP_VAR, 153
 MSK_RES_ERR_CONE_SIZE, 153
 MSK_RES_ERR_CONE_TYPE, 153
 MSK_RES_ERR_CONE_TYPE_STR, 153
 MSK_RES_ERR_DATA_FILE_EXT, 147
 MSK_RES_ERR_DIMENSION_SPECIFICATION, 148
 MSK_RES_ERR_DJC_AFE_DOMAIN_MISMATCH, 162
 MSK_RES_ERR_DJC_DOMAIN_TERMSIZE_MISMATCH, 162
 MSK_RES_ERR_DJC_INVALID_INDEX, 162
 MSK_RES_ERR_DJC_INVALID_TERM_SIZE, 162
 MSK_RES_ERR_DJC_TOTAL_NUM_TERMS_MISMATCH, 162
 MSK_RES_ERR_DJC_UNSUPPORTED_DOMAIN_TYPE, 162
 MSK_RES_ERR_DOMAIN_DIMENSION, 161
 MSK_RES_ERR_DOMAIN_DIMENSION_PSD, 161
 MSK_RES_ERR_DOMAIN_INVALID_INDEX, 161
 MSK_RES_ERR_DOMAIN_POWER_INVALID_ALPHA, 161
 MSK_RES_ERR_DOMAIN_POWER_NEGATIVE_ALPHA, 161
 MSK_RES_ERR_DOMAIN_POWER_NLEFT, 161
 MSK_RES_ERR_DUP_NAME, 147
 MSK_RES_ERR_DUPLICATE_AIJ, 154
 MSK_RES_ERR_DUPLICATE_BARVARIABLE_NAMES, 158
 MSK_RES_ERR_DUPLICATE_CONE_NAMES, 158
 MSK_RES_ERR_DUPLICATE_CONSTRAINT_NAMES, 158
 MSK_RES_ERR_DUPLICATE_DJC_NAMES, 158
 MSK_RES_ERR_DUPLICATE_DOMAIN_NAMES, 158
 MSK_RES_ERR_DUPLICATE_FIJ, 161
 MSK_RES_ERR_DUPLICATE_VARIABLE_NAMES, 158
 MSK_RES_ERR_END_OF_FILE, 147
 MSK_RES_ERR_FACTOR, 155
 MSK_RES_ERR_FEASREPAIR_CANNOT_RELAX, 156
 MSK_RES_ERR_FEASREPAIR_INCONSISTENT_BOUND, 156
 MSK_RES_ERR_FEASREPAIR_SOLVING_RELAXED, 156
 MSK_RES_ERR_FILE_LICENSE, 145
 MSK_RES_ERR_FILE_OPEN, 147
 MSK_RES_ERR_FILE_READ, 147
 MSK_RES_ERR_FILE_WRITE, 147
 MSK_RES_ERR_FINAL_SOLUTION, 155
 MSK_RES_ERR_FIRST, 155
 MSK_RES_ERR_FIRSTI, 153
 MSK_RES_ERR_FIRSTJ, 153
 MSK_RES_ERR_FIXED_BOUND_VALUES, 154
 MSK_RES_ERR_FLEXLM, 146
 MSK_RES_ERR_FORMAT_STRING, 147
 MSK_RES_ERR_GLOBAL_INV_CONIC_PROBLEM, 155
 MSK_RES_ERR_HUGE_AIJ, 154
 MSK_RES_ERR_HUGE_C, 154
 MSK_RES_ERR_HUGE_FIJ, 161
 MSK_RES_ERR_IDENTICAL_TASKS, 157
 MSK_RES_ERR_IN_ARGUMENT, 150
 MSK_RES_ERR_INDEX, 151
 MSK_RES_ERR_INDEX_ARR_IS_TOO_LARGE, 151
 MSK_RES_ERR_INDEX_ARR_IS_TOO_SMALL, 151
 MSK_RES_ERR_INDEX_IS_NOT_UNIQUE, 150
 MSK_RES_ERR_INDEX_IS_TOO_LARGE, 150
 MSK_RES_ERR_INDEX_IS_TOO_SMALL, 150
 MSK_RES_ERR_INF_DOU_INDEX, 151
 MSK_RES_ERR_INF_DOU_NAME, 151
 MSK_RES_ERR_INF_IN_DOUBLE_DATA, 154
 MSK_RES_ERR_INF_INT_INDEX, 151
 MSK_RES_ERR_INF_INT_NAME, 151
 MSK_RES_ERR_INF_LINT_INDEX, 151
 MSK_RES_ERR_INF_LINT_NAME, 151
 MSK_RES_ERR_INF_TYPE, 151
 MSK_RES_ERR_INFEAS_UNDEFINED, 157
 MSK_RES_ERR_INFINITE_BOUND, 154
 MSK_RES_ERR_INT64_TO_INT32_CAST, 157
 MSK_RES_ERR_INTERNAL, 156
 MSK_RES_ERR_INTERNAL_TEST_FAILED, 157
 MSK_RES_ERR_INV_APTRE, 152
 MSK_RES_ERR_INV_BK, 152
 MSK_RES_ERR_INV_BKC, 152
 MSK_RES_ERR_INV_BKX, 152
 MSK_RES_ERR_INV_CONE_TYPE, 152
 MSK_RES_ERR_INV_CONE_TYPE_STR, 152
 MSK_RES_ERR_INV_MARKI, 156
 MSK_RES_ERR_INV_MARKJ, 156
 MSK_RES_ERR_INV_NAME_ITEM, 152
 MSK_RES_ERR_INV_NUMI, 156
 MSK_RES_ERR_INV_NUMJ, 156
 MSK_RES_ERR_INV_OPTIMIZER, 155
 MSK_RES_ERR_INV_PROBLEM, 155
 MSK_RES_ERR_INV_QCON_SUBI, 154
 MSK_RES_ERR_INV_QCON_SUBJ, 154
 MSK_RES_ERR_INV_QCON_SUBK, 154
 MSK_RES_ERR_INV_QCON_VAL, 154
 MSK_RES_ERR_INV_QOBJ_SUBI, 154
 MSK_RES_ERR_INV_QOBJ_SUBJ, 154
 MSK_RES_ERR_INV_QOBJ_VAL, 154
 MSK_RES_ERR_INV_SK, 152
 MSK_RES_ERR_INV_SK_STR, 152
 MSK_RES_ERR_INV_SKC, 152

MSK_RES_ERR_INV_SKN, 152
 MSK_RES_ERR_INV_SKX, 152
 MSK_RES_ERR_INV_VAR_TYPE, 152
 MSK_RES_ERR_INVALID_AIJ, 155
 MSK_RES_ERR_INVALID_AMPL_STUB, 157
 MSK_RES_ERR_INVALID_B, 161
 MSK_RES_ERR_INVALID_BARVAR_NAME, 147
 MSK_RES_ERR_INVALID_CFIX, 155
 MSK_RES_ERR_INVALID_CJ, 155
 MSK_RES_ERR_INVALID_COMPRESSION, 156
 MSK_RES_ERR_INVALID_CON_NAME, 147
 MSK_RES_ERR_INVALID_CONE_NAME, 147
 MSK_RES_ERR_INVALID_FIJ, 161
 MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_AFFINE_CONIC_PROBLEMS, 158
 MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_CFIX, 158
 MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_CONES, 158
 MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_DISJUNCTIVE CONSTRAINTS, 158
 MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_FREE_CONSTRAINTS, 158
 MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_NONLINEAR_CONSTRAINTS, 158
 MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_QUADRATIC_CONSTRAINTS, 158
 MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_RANGED_CONSTRAINTS, 158
 MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_SYM_MAT, 158
 MSK_RES_ERR_INVALID_FILE_NAME, 147
 MSK_RES_ERR_INVALID_FORMAT_TYPE, 153
 MSK_RES_ERR_INVALID_G, 161
 MSK_RES_ERR_INVALID_IDX, 152
 MSK_RES_ERR_INVALID_IOMODE, 156
 MSK_RES_ERR_INVALID_MAX_NUM, 152
 MSK_RES_ERR_INVALID_NAME_IN_SOL_FILE, 150
 MSK_RES_ERR_INVALID_OBJ_NAME, 147
 MSK_RES_ERR_INVALID_OBJECTIVE_SENSE, 154
 MSK_RES_ERR_INVALID_PROBLEM_TYPE, 158
 MSK_RES_ERR_INVALID_SOL_FILE_NAME, 147
 MSK_RES_ERR_INVALID_STREAM, 147
 MSK_RES_ERR_INVALID_SURPLUS, 152
 MSK_RES_ERR_INVALID_SYM_MAT_DIM, 158
 MSK_RES_ERR_INVALID_TASK, 147
 MSK_RES_ERR_INVALID_UTF8, 156
 MSK_RES_ERR_INVALID_VAR_NAME, 147
 MSK_RES_ERR_INVALID_WCHAR, 156
 MSK_RES_ERR_INVALID_WHICH_SOL, 151
 MSK_RES_ERR_JSON_DATA, 150
 MSK_RES_ERR_JSON_FORMAT, 150
 MSK_RES_ERR_JSON_MISSING_DATA, 150
 MSK_RES_ERR_JSON_NUMBER_OVERFLOW, 150
 MSK_RES_ERR_JSON_STRING, 150
 MSK_RES_ERR_JSON_SYNTAX, 150
 MSK_RES_ERR_LAST, 155
 MSK_RES_ERR_LASTI, 153
 MSK_RES_ERR_LASTJ, 153
 MSK_RES_ERR_LAU_ARG_K, 159
 MSK_RES_ERR_LAU_ARG_M, 159
 MSK_RES_ERR_LAU_ARG_N, 159
 MSK_RES_ERR_LAU_ARG_TRANS, 159
 MSK_RES_ERR_LAU_ARG_TRANSA, 159
 MSK_RES_ERR_LAU_ARG_TRANSB, 159
 MSK_RES_ERR_LAU_ARG_UPLO, 159
 MSK_RES_ERR_LAU_INVALID_LOWER_TRIANGULAR_MATRIX, 158
 MSK_RES_ERR_LAU_INVALID_SPARSE_SYMMETRIC_MATRIX, 159
 MSK_RES_ERR_LAU_NOT_POSITIVE_DEFINITE, 158
 MSK_RES_ERR_LAU_INVALID_LOWER_TRIANGULAR_MATRIX, 158
 MSK_RES_ERR_LAU_UNKNOWN, 158
 MSK_RES_ERR_LICENSE, 145
 MSK_RES_ERR_LICENSE_CANNOT_ALLOCATE, 146
 MSK_RES_ERR_LICENSE_CANNOT_CONNECT, 146
 MSK_RES_ERR_LICENSE_EXPIRED, 145
 MSK_RES_ERR_LICENSE_FEATURE, 146
 MSK_RES_ERR_LICENSE_INVALID_HOSTID, 146
 MSK_RES_ERR_LICENSE_MAX, 146
 MSK_RES_ERR_LICENSE_MOSEKLM_DAEMON, 146
 MSK_RES_ERR_LICENSE_NO_SERVER_LINE, 146
 MSK_RES_ERR_LICENSE_NO_SERVER_SUPPORT, 146
 MSK_RES_ERR_LICENSE_OLD_SERVER_VERSION, 145
 MSK_RES_ERR_LICENSE_SERVER, 146
 MSK_RES_ERR_LICENSE_SERVER_VERSION, 146
 MSK_RES_ERR_LICENSE_VERSION, 145
 MSK_RES_ERR_LINK_FILE_DLL, 146
 MSK_RES_ERR_LIVING_TASKS, 147
 MSK_RES_ERR_LOWER_BOUND_IS_A_NAN, 154
 MSK_RES_ERR_LP_DUP_SLACK_NAME, 149
 MSK_RES_ERR_LP_EMPTY, 149
 MSK_RES_ERR_LP_FILE_FORMAT, 149
 MSK_RES_ERR_LP_FREE_CONSTRAINT, 149
 MSK_RES_ERR_LP_INCOMPATIBLE, 149
 MSK_RES_ERR_LP_INDICATOR_VAR, 150
 MSK_RES_ERR_LP_INVALID_CON_NAME, 150
 MSK_RES_ERR_LP_INVALID_VAR_NAME, 149
 MSK_RES_ERR_LP_WRITE_CONIC_PROBLEM, 150
 MSK_RES_ERR_LP_WRITE_GECO_PROBLEM, 150
 MSK_RES_ERR_LU_MAX_NUM_TRIES, 156
 MSK_RES_ERR_MAX_LEN_IS_TOO_SMALL, 153
 MSK_RES_ERR_MAXNUMBARVAR, 151
 MSK_RES_ERR_MAXNUMCON, 151
 MSK_RES_ERR_MAXNUMCONE, 153
 MSK_RES_ERR_MAXNUMQNZ, 152
 MSK_RES_ERR_MAXNUMVAR, 151
 MSK_RES_ERR_MIO_INTERNAL, 158
 MSK_RES_ERR_MIO_INVALID_NODE_OPTIMIZER, 160
 MSK_RES_ERR_MIO_INVALID_ROOT_OPTIMIZER, 160
 MSK_RES_ERR_MIO_NO_OPTIMIZER, 155
 MSK_RES_ERR_MISMATCHING_DIMENSION, 147
 MSK_RES_ERR_MISSING_LICENSE_FILE, 146
 MSK_RES_ERR_MIXED_CONIC_AND_NL, 155
 MSK_RES_ERR_MPS_CONE_OVERLAP, 148
 MSK_RES_ERR_MPS_CONE_REPEAT, 148

MSK_RES_ERR_MPS_CONE_TYPE, 148
 MSK_RES_ERR_MPS_DUPLICATE_Q_ELEMENT, 148
 MSK_RES_ERR_MPS_FILE, 148
 MSK_RES_ERR_MPS_INV_FIELD, 148
 MSK_RES_ERR_MPS_INV_MARKER, 148
 MSK_RES_ERR_MPS_INV_SEC_ORDER, 148
 MSK_RES_ERR_MPS_INVALID_BOUND_KEY, 148
 MSK_RES_ERR_MPS_INVALID_CON_KEY, 148
 MSK_RES_ERR_MPS_INVALID_INDICATOR_CONSTRAINT, 149
 MSK_RES_ERR_MPS_INVALID_INDICATOR_QUADRATIC, 149
 MSK_RES_ERR_MPS_INVALID_INDICATOR_VALUE, 149
 MSK_RES_ERR_MPS_INVALID_INDICATOR_VARIABLE, 149
 MSK_RES_ERR_MPS_INVALID_KEY, 149
 MSK_RES_ERR_MPS_INVALID_OBJ_NAME, 149
 MSK_RES_ERR_MPS_INVALID_OBJSENSE, 148
 MSK_RES_ERR_MPS_INVALID_SEC_NAME, 148
 MSK_RES_ERR_MPS_MUL_CON_NAME, 148
 MSK_RES_ERR_MPS_MUL_CSEC, 148
 MSK_RES_ERR_MPS_MUL_QOBJ, 148
 MSK_RES_ERR_MPS_MUL_QSEC, 148
 MSK_RES_ERR_MPS_NO_OBJECTIVE, 148
 MSK_RES_ERR_MPS_NON_SYMMETRIC_Q, 148
 MSK_RES_ERR_MPS_NULL_CON_NAME, 148
 MSK_RES_ERR_MPS_NULL_VAR_NAME, 148
 MSK_RES_ERR_MPS_SPLITTED_VAR, 148
 MSK_RES_ERR_MPS_TAB_IN_FIELD2, 149
 MSK_RES_ERR_MPS_TAB_IN_FIELD3, 149
 MSK_RES_ERR_MPS_TAB_IN_FIELD5, 149
 MSK_RES_ERR_MPS_UNDEF_CON_NAME, 148
 MSK_RES_ERR_MPS_UNDEF_VAR_NAME, 148
 MSK_RES_ERR_MPS_WRITE_CPLEX_INVALID_CONE_TYPE, 160
 MSK_RES_ERR_MUL_A_ELEMENT, 152
 MSK_RES_ERR_NAME_IS_NULL, 156
 MSK_RES_ERR_NAME_MAX_LEN, 156
 MSK_RES_ERR_NAN_IN_BLC, 155
 MSK_RES_ERR_NAN_IN_BLX, 155
 MSK_RES_ERR_NAN_IN_BUC, 155
 MSK_RES_ERR_NAN_IN_BUX, 155
 MSK_RES_ERR_NAN_IN_C, 155
 MSK_RES_ERR_NAN_IN_DOUBLE_DATA, 154
 MSK_RES_ERR_NEGATIVE_APPEND, 155
 MSK_RES_ERR_NEGATIVE_SURPLUS, 155
 MSK_RES_ERR_NEWER_DLL, 146
 MSK_RES_ERR_NO_BARS_FOR_SOLUTION, 157
 MSK_RES_ERR_NO_BARX_FOR_SOLUTION, 157
 MSK_RES_ERR_NO_BASIS_SOL, 155
 MSK_RES_ERR_NO_DOTY, 162
 MSK_RES_ERR_NO_DUAL_FOR_ITG_SOL, 156
 MSK_RES_ERR_NO_DUAL_INFEAS_CER, 156
 MSK_RES_ERR_NO_INIT_ENV, 147
 MSK_RES_ERR_NO_OPTIMIZER_VAR_TYPE, 155
 MSK_RES_ERR_NO_PRIMAL_INFEAS_CER, 156
 MSK_RES_ERR_NO_SNX_FOR_BAS_SOL, 156
 MSK_RES_ERR_NO_SOLUTION_IN_CALLBACK, 156
 MSK_RES_ERR_NON_UNIQUE_ARRAY, 158
 MSK_RES_ERR_NONCONVEX, 153
 MSK_RES_ERR_NONLINEAR_EQUALITY, 153
 MSK_RES_ERR_NONLINEAR_RANGED, 153
 MSK_RES_ERR_NOT_POWER_DOMAIN, 161
 MSK_RES_ERR_NULL_ENV, 147
 MSK_RES_ERR_NULL_POINTER, 147
 MSK_RES_ERR_NULL_TASK, 147
 MSK_RES_ERR_NUM_ARGUMENTS, 150
 MSK_RES_ERR_NUMCONLIM, 152
 MSK_RES_ERR_NUMVARLIM, 152
 MSK_RES_ERR_OBJ_Q_NOT_NSD, 153
 MSK_RES_ERR_OBJ_Q_NOT_PSD, 153
 MSK_RES_ERR_OBJECTIVE_RANGE, 152
 MSK_RES_ERR_OLDER_DLL, 146
 MSK_RES_ERR_OPF_DUAL_INTEGER_SOLUTION, 149
 MSK_RES_ERR_OPF_DUPLICATE_BOUND, 149
 MSK_RES_ERR_OPF_DUPLICATE_CONE_ENTRY, 149
 MSK_RES_ERR_OPF_DUPLICATE_CONSTRAINT_NAME, 149
 MSK_RES_ERR_OPF_INCORRECT_TAG_PARAM, 149
 MSK_RES_ERR_OPF_INVALID_CONE_TYPE, 149
 MSK_RES_ERR_OPF_INVALID_TAG, 149
 MSK_RES_ERR_OPF_MISMATCHED_TAG, 149
 MSK_RES_ERR_OPF_PREMATURE_EOF, 149
 MSK_RES_ERR_OPF_SYNTAX, 149
 MSK_RES_ERR_OPF_TOO_LARGE, 149
 MSK_RES_ERR_OPTIMIZER_LICENSE, 146
 MSK_RES_ERR_OVERFLOW, 155
 MSK_RES_ERR_PARAM_INDEX, 151
 MSK_RES_ERR_PARAM_IS_TOO_LARGE, 151
 MSK_RES_ERR_PARAM_IS_TOO_SMALL, 151
 MSK_RES_ERR_PARAM_NAME, 151
 MSK_RES_ERR_PARAM_NAME_DOU, 151
 MSK_RES_ERR_PARAM_NAME_INT, 151
 MSK_RES_ERR_PARAM_NAME_STR, 151
 MSK_RES_ERR_PARAM_TYPE, 151
 MSK_RES_ERR_PARAM_VALUE_STR, 151
 MSK_RES_ERR_PLATFORM_NOT_LICENSED, 146
 MSK_RES_ERR_POSTSOLVE, 155
 MSK_RES_ERR_PRO_ITEM, 152
 MSK_RES_ERR_PROB_LICENSE, 145
 MSK_RES_ERR_PTF_FORMAT, 150
 MSK_RES_ERR_PTF_INCOMPATIBILITY, 150
 MSK_RES_ERR_PTF_INCONSISTENCY, 150
 MSK_RES_ERR_PTF_UNDEFINED_ITEM, 150
 MSK_RES_ERR_QCON_SUBI_TOO_LARGE, 154
 MSK_RES_ERR_QCON_SUBI_TOO_SMALL, 154
 MSK_RES_ERR_QCON_UPPER_TRIANGLE, 154
 MSK_RES_ERR_QOBJ_UPPER_TRIANGLE, 154
 MSK_RES_ERR_READ_FORMAT, 148
 MSK_RES_ERR_READ_LP_MISSING_END_TAG, 150
 MSK_RES_ERR_READ_LP_NONEXISTING_NAME, 150
 MSK_RES_ERR_REMOVE_CONE_VARIABLE, 154
 MSK_RES_ERR_REPAIR_INVALID_PROBLEM, 156
 MSK_RES_ERR_REPAIR_OPTIMIZATION_FAILED, 156
 MSK_RES_ERR_SEN_BOUND_INVALID_LO, 157

MSK_RES_ERR_SEN_BOUND_INVALID_UP, 157
 MSK_RES_ERR_SEN_FORMAT, 157
 MSK_RES_ERR_SEN_INDEX_INVALID, 157
 MSK_RES_ERR_SEN_INDEX_RANGE, 157
 MSK_RES_ERR_SEN_INVALID_REGEX, 157
 MSK_RES_ERR_SEN_NUMERICAL, 157
 MSK_RES_ERR_SEN_SOLUTION_STATUS, 157
 MSK_RES_ERR_SEN_UNDEF_NAME, 157
 MSK_RES_ERR_SEN_UNHANDLED_PROBLEM_TYPE, 157
 MSK_RES_ERR_SERVER_ACCESS_TOKEN, 161
 MSK_RES_ERR_SERVER_ADDRESS, 161
 MSK_RES_ERR_SERVER_CERTIFICATE, 161
 MSK_RES_ERR_SERVER_CONNECT, 161
 MSK_RES_ERR_SERVER_PROBLEM_SIZE, 161
 MSK_RES_ERR_SERVER_PROTOCOL, 161
 MSK_RES_ERR_SERVER_STATUS, 161
 MSK_RES_ERR_SERVER_TLS_CLIENT, 161
 MSK_RES_ERR_SERVER_TOKEN, 161
 MSK_RES_ERR_SHAPE_IS_TOO_LARGE, 150
 MSK_RES_ERR_SIZE_LICENSE, 145
 MSK_RES_ERR_SIZE_LICENSE_CON, 146
 MSK_RES_ERR_SIZE_LICENSE_INTVAR, 146
 MSK_RES_ERR_SIZE_LICENSE_NUMCORES, 157
 MSK_RES_ERR_SIZE_LICENSE_VAR, 146
 MSK_RES_ERR_SLICE_SIZE, 155
 MSK_RES_ERR_SOL_FILE_INVALID_NUMBER, 154
 MSK_RES_ERR_SOLITEM, 151
 MSK_RES_ERR_SOLVER_PROBTYPE, 152
 MSK_RES_ERR_SPACE, 147
 MSK_RES_ERR_SPACE_LEAKING, 148
 MSK_RES_ERR_SPACE_NO_INFO, 148
 MSK_RES_ERR_SPARSITY_SPECIFICATION, 147
 MSK_RES_ERR_SYM_MAT_DUPLICATE, 158
 MSK_RES_ERR_SYM_MAT_HUGE, 155
 MSK_RES_ERR_SYM_MAT_INVALID, 155
 MSK_RES_ERR_SYM_MAT_INVALID_COL_INDEX, 157
 MSK_RES_ERR_SYM_MAT_INVALID_ROW_INDEX, 157
 MSK_RES_ERR_SYM_MAT_INVALID_VALUE, 158
 MSK_RES_ERR_SYM_MAT_NOT_LOWER_TRINGULAR,
 157
 MSK_RES_ERR_TASK_INCOMPATIBLE, 156
 MSK_RES_ERR_TASK_INVALID, 156
 MSK_RES_ERR_TASK_WRITE, 156
 MSK_RES_ERR_THREAD_COND_INIT, 147
 MSK_RES_ERR_THREAD_CREATE, 146
 MSK_RES_ERR_THREAD_MUTEX_INIT, 146
 MSK_RES_ERR_THREAD_MUTEX_LOCK, 146
 MSK_RES_ERR_THREAD_MUTEX_UNLOCK, 146
 MSK_RES_ERR_TOCONIC_CONSTR_NOT_CONIC, 161
 MSK_RES_ERR_TOCONIC_CONSTR_Q_NOT_PSD, 161
 MSK_RES_ERR_TOCONIC_CONSTRAINT_FX, 161
 MSK_RES_ERR_TOCONIC_CONSTRAINT_RA, 161
 MSK_RES_ERR_TOCONIC_OBJECTIVE_NOT_PSD, 161
 MSK_RES_ERR_TOO_SMALL_A_TRUNCATION_VALUE,
 154
 MSK_RES_ERR_TOO_SMALL_MAX_NUM_NZ, 152
 MSK_RES_ERR_TOO_SMALL_MAXNUMANZ, 152
 MSK_RES_ERR_UNALLOWED_WHICHSOL, 152
 MSK_RES_ERR_UNB_STEP_SIZE, 157
 MSK_RES_ERR_UNDEF_SOLUTION, 162
 MSK_RES_ERR_UNDEFINED_OBJECTIVE_SENSE, 154
 MSK_RES_ERR_UNHANDLED_SOLUTION_STATUS, 158
 MSK_RES_ERR_UNKNOWN, 147
 MSK_RES_ERR_UPPER_BOUND_IS_A_NAN, 154
 MSK_RES_ERR_UPPER_TRIANGLE, 158
 MSK_RES_ERR_WHICHITEM_NOT_ALLOWED, 151
 MSK_RES_ERR_WHICHSOL, 151
 MSK_RES_ERR_WRITE_LP_FORMAT, 150
 MSK_RES_ERR_WRITE_LP_NON_UNIQUE_NAME, 150
 MSK_RES_ERR_WRITE_MPS_INVALID_NAME, 149
 MSK_RES_ERR_WRITE_OPF_INVALID_VAR_NAME, 149
 MSK_RES_ERR_WRITING_FILE, 150
 MSK_RES_ERR_XML_INVALID_PROBLEM_TYPE, 157
 MSK_RES_ERR_Y_IS_UNDEFINED, 154

Index

Symbols

`--`
 mosek command line option, 18

`-?`
 mosek command line option, 17

`-a`
 mosek command line option, 16

`-anapro`
 mosek command line option, 16

`-anasoli <name>`
 mosek command line option, 16

`-anasolo <name>`
 mosek command line option, 16

`-basi <name>`
 mosek command line option, 16

`-baso <name>`
 mosek command line option, 16

`-d <name> <value>`
 mosek command line option, 16

`-dbgmem <name>`
 mosek command line option, 16

`-f`
 mosek command line option, 17

`-h`
 mosek command line option, 17

`-info <name>`
 mosek command line option, 17

`-infrepo <name>`
 mosek command line option, 17

`-inti <name>`
 mosek command line option, 17

`-into <name>`
 mosek command line option, 17

`-itri <name>`
 mosek command line option, 17

`-itro <name>`
 mosek command line option, 17

`-jsoli <name>`
 mosek command line option, 17

`-jsolo <name>`
 mosek command line option, 17

`-l, -L <dir>`
 mosek command line option, 17

`-max`
 mosek command line option, 17

`-min`
 mosek command line option, 17

`-n`
 mosek command line option, 17

`-optserv <url>`
 mosek command line option, 17

`-out <name>`
 mosek command line option, 17

`-p <name>`
 mosek command line option, 17

`-pari <name>`
 mosek command line option, 17

`-paro <name>`
 mosek command line option, 17

`-primalrepair`
 mosek command line option, 17

`-q <name>`
 mosek command line option, 17

`-r`
 mosek command line option, 17

`-removeitg`
 mosek command line option, 17

`-rout <name>`
 mosek command line option, 17

`-sen <file>`
 mosek command line option, 17

`-silent`
 mosek command line option, 17

`-toconic`
 mosek command line option, 17

`-v`
 mosek command line option, 17

`-w`
 mosek command line option, 17

`-x`
 mosek command line option, 17

A

AMPL
 outlev, 22
 wantsol, 22

analysis
 infeasibility, 78

arguments
 command line tool, 12

B

basis identification, 58

basis type
 sensitivity analysis, 84

big-M, 74

bound
 constraint, 43, 46, 50

variable, 43, 46, 50
Branch-and-Bound, 66

C

CBF format, 216
certificate
 dual, 45, 48
 primal, 45, 48
command line tool
 arguments, 12
complementarity, 44, 48
cone
 dual, 47
conic optimization, 46
 interior-point, 62
 mixed-integer, 73
 termination criteria, 64
constraint
 bound, 43, 46, 50
 matrix, 43, 46, 50
 quadratic, 51
cuts, 71
cutting planes, 71

D

disjunctive constraint, 74
domain, 187
dual
 certificate, 45, 48
 cone, 47
 feasible, 44
 infeasible, 44, 45, 48
 problem, 43, 47, 51
 variable, 43, 47
duality
 conic, 47
 linear, 43
 semidefinite, 51
dualizer, 54

E

eliminator, 54

F

feasibility
 integer feasibility, 69
feasible
 dual, 44
 primal, 43, 56, 63
 problem, 43
format
 CBF, 216
 json, 241
 LP, 191
 MPS, 195
 OPF, 207
 PTF, 234
 sol, 247

task, 240

H

heuristic, 70
hot-start, 60

I

infeasibility, 45, 48
 analysis, 78
 linear optimization, 45
 repair, 78
 semidefinite, 51
infeasible
 dual, 44, 45, 48
 primal, 43, 45, 48, 56, 63
 problem, 43, 45, 51
installation, 9
 requirements, 9
 troubleshooting, 9
integer feasibility, 69
 feasibility, 69
interior-point
 conic optimization, 62
 linear optimization, 56
 logging, 59, 65
 optimizer, 56, 62
 termination criteria, 57, 64

J

json format, 241

L

license, 18
linear dependency, 54
linear optimization, 43
 infeasibility, 45
 interior-point, 56
 simplex, 60
 termination criteria, 57, 60
linearity interval, 82
logging
 interior-point, 59, 65
 mixed-integer optimizer, 72
 optimizer, 59, 61, 65
 simplex, 61
LP format, 191

M

matrix
 constraint, 43, 46, 50
MI(QC)QO, 73
MICO, 73
MIP, *see* integer optimization
mixed-integer, *see* integer
 conic optimization, 73
 optimizer, 66
 presolve, 70
 quadratic, 73

- mixed-integer optimization, 66
- mixed-integer optimizer
 - logging, 72
- mosek command line option
 - `==`, 18
 - `??`, 17
 - `-a`, 16
 - `-anapro`, 16
 - `-anasoli <name>`, 16
 - `-anasolo <name>`, 16
 - `-basi <name>`, 16
 - `-baso <name>`, 16
 - `-d <name> <value>`, 16
 - `-dbgmem <name>`, 16
 - `-f`, 17
 - `-h`, 17
 - `-info <name>`, 17
 - `-infrepo <name>`, 17
 - `-inti <name>`, 17
 - `-into <name>`, 17
 - `-itri <name>`, 17
 - `-itro <name>`, 17
 - `-jsoli <name>`, 17
 - `-jsolo <name>`, 17
 - `-l, -L <dir>`, 17
 - `-max`, 17
 - `-min`, 17
 - `-n`, 17
 - `-optserv <url>`, 17
 - `-out <name>`, 17
 - `-p <name>`, 17
 - `-pari <name>`, 17
 - `-paro <name>`, 17
 - `-primalrepair`, 17
 - `-q <name>`, 17
 - `-r`, 17
 - `-removeitg`, 17
 - `-rout <name>`, 17
 - `-sen <file>`, 17
 - `-silent`, 17
 - `-toconic`, 17
 - `-v`, 17
 - `-w`, 17
 - `-x`, 17
- MPS format, 195
 - free, 206

N

- numerical issues
 - presolve, 54
 - scaling, 55
 - simplex, 61

O

- objective, 43, 46, 50
- OPF format, 207
- optimality gap, 68
- optimization

- conic, 46
- conic quadratic, 46
- linear, 43
- semidefinite, 49
- optimizer
 - interior-point, 56, 62
 - logging, 59, 61, 65
 - mixed-integer, 66
 - selection, 54, 55
 - simplex, 60
 - termination, 68
- outlev
 - AMPL, 22

P

- parameter
 - simplex, 61
- parameter file, 16
- presolve, 53
 - eliminator, 54
 - linear dependency check, 54
 - mixed-integer, 70
 - numerical issues, 54
- primal
 - certificate, 45, 48
 - feasible, 43, 56, 63
 - infeasible, 43, 45, 48, 56, 63
 - problem, 43, 47, 51
 - solution, 43
- primal heuristics, 70
- primal-dual
 - problem, 56, 62
 - solution, 44
- problem
 - dual, 43, 47, 51
 - feasible, 43
 - infeasible, 43, 45, 51
 - primal, 43, 47, 51
 - primal-dual, 56, 62
 - unbounded, 45, 49
- PTF format, 234

Q

- quadratic
 - constraint, 51
 - mixed-integer, 73
- quadratic optimization, 51

R

- relaxation, 66
- repair
 - infeasibility, 78

S

- scaling, 55
- semidefinite
 - infeasibility, 51
- semidefinite optimization, 49

- sensitivity analysis, 82
 - basis type, 84
- shadow price, 82
- simplex
 - linear optimization, 60
 - logging, 61
 - numerical issues, 61
 - optimizer, 60
 - parameter, 61
 - termination criteria, 60
- sol format, 247
- solution
 - file format, 247
 - primal, 43
 - primal-dual, 44

T

- task format, 240
- termination
 - optimizer, 68
- termination criteria, 68
 - conic optimization, 64
 - interior-point, 57, 64
 - linear optimization, 57, 60
 - simplex, 60
 - tolerance, 58, 65
- tolerance
 - termination criteria, 58, 65
- troubleshooting
 - installation, 9

U

- unbounded
 - problem, 45, 49

V

- valid inequalities, 71
- variable, 43, 46, 50
 - bound, 43, 46, 50
 - dual, 43, 47

W

- wantsol
 - AMPL, 22