# MOSEK Optimization Toolbox for MATLAB

*Release 8.1.0.53*

MOSEK ApS

2018

# CONTENTS


**1 Introduction** **1**
1.1 Why the Optimization Toolbox for MATLAB? . . . . . 2

**2 Contact Information** **3**

**3 License Agreement** **5**

**4 Installation** **7**
4.1 Testing the installation . . . . . 8

**5 Guidelines** **11**
5.1 The **MOSEK** integration with MATLAB . . . . . 11
5.2 Caveats Using the MATLAB Compiler . . . . . 12
5.3 The license system . . . . . 12

**6 Basic Tutorials** **13**
6.1 The Basics Tutorial . . . . . 14
6.2 Linear Optimization . . . . . 14
6.3 Conic Quadratic Optimization . . . . . 18
6.4 Semidefinite Optimization . . . . . 19
6.5 Quadratic Optimization . . . . . 21
6.6 Integer Optimization . . . . . 25
6.7 Optimizer Termination Handling . . . . . 26
6.8 Problem Modification and Reoptimization . . . . . 28
6.9 Solution Analysis . . . . . 31
6.10 Solver Parameters . . . . . 35

**7 Nonlinear Tutorials** **37**
7.1 Separable Convex (SCopt) Interface . . . . . 37
7.2 Entropy Optimization . . . . . 39
7.3 Geometric Optimization . . . . . 40

**8 Advanced Tutorials** **43**
8.1 Linear Least Squares and Related Norm Minimization Problems . . . . . 43
8.2 Converting a quadratically constrained problem to conic form . . . . . 50

**9 Case Studies** **55**
9.1 Robust linear Optimization . . . . . 55
9.2 Geometric (posynomial) Optimization . . . . . 67

**10 Managing I/O** **73**
10.1 Stream I/O . . . . . 73
10.2 File I/O . . . . . 73
10.3 Verbosity . . . . . 74

**11 The Optimizers for Continuous Problems** **75**
11.1 Presolve . . . 75
11.2 Using Multiple Threads in an Optimizer . . . 77
11.3 Linear Optimization . . . 78
11.4 Conic Optimization . . . 85
11.5 Nonlinear Convex Optimization . . . 89

**12 The Optimizer for Mixed-integer Problems** **91**
12.1 The Mixed-integer Optimizer Overview . . . 91
12.2 Relaxations and bounds . . . 91
12.3 Termination Criterion . . . 92
12.4 Speeding Up the Solution Process . . . 93
12.5 Understanding Solution Quality . . . 93
12.6 The Optimizer Log . . . 94

**13 Problem Analyzer** **95**

**14 Analyzing Infeasible Problems** **99**
14.1 Example: Primal Infeasibility . . . 99
14.2 Locating the cause of Primal Infeasibility . . . 100
14.3 Locating the Cause of Dual Infeasibility . . . 101
14.4 The Infeasibility Report . . . 101
14.5 Theory Concerning Infeasible Problems . . . 103
14.6 The Certificate of Primal Infeasibility . . . 103
14.7 The certificate of dual infeasibility . . . 104

**15 Sensitivity Analysis** **109**
15.1 Sensitivity Analysis for Linear Problems . . . 109
15.2 Sensitivity Analysis with **MOSEK** . . . 115

**16 Problem Formulation and Solutions** **121**
16.1 Linear Optimization . . . 121
16.2 Conic Quadratic Optimization . . . 124
16.3 Semidefinite Optimization . . . 126
16.4 Quadratic and Quadratically Constrained Optimization . . . 128
16.5 General Convex Optimization . . . 129

**17 Toolbox Reference** **131**
17.1 Command Reference . . . 131
17.2 Data Structures and Notation . . . 142
17.3 Parameters grouped by topic . . . 151
17.4 Parameters (alphabetical list sorted by type) . . . 162
17.5 Response codes . . . 199
17.6 Enumerations . . . 221

**18 Supported File Formats** **247**
18.1 The LP File Format . . . 248
18.2 The MPS File Format . . . 253
18.3 The OPF Format . . . 265
18.4 The CBF Format . . . 274
18.5 The XML (OSiL) Format . . . 289
18.6 The Task Format . . . 289
18.7 The JSON Format . . . 290
18.8 The Solution File Format . . . 297

**19 List of examples** **301**

**20 Interface changes** **303**
20.1 Compatibility . . . 303
20.2 Parameters . . . 303

20.3 Constants . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 306
20.4 Response Codes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 310

**Bibliography** **313**

**Symbol Index** **315**

**Index** **327**

CHAPTER

ONE

# INTRODUCTION

The **MOSEK** Optimization Suite 8.1.0.53 is a powerful software package capable of solving large-scale optimization problems of the following kind:

- linear,
- conic quadratic (also known as second-order cone),
- convex quadratic,
- semidefinite,
- and general convex.

Integer constrained variables are supported for all problem classes except for semidefinite and general convex problems. 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

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

where $\mathcal{K} = \{y : y \geq 0\}$, i.e.,

$$\begin{aligned} &Ax - b = y,\\ &y \in \mathcal{K}. \end{aligned}$$

In conic optimization a wider class of convex sets $\mathcal{K}$ is allowed, for example in 3 dimensions $\mathcal{K}$ may correspond to an ice cream cone. The conic optimizer in **MOSEK** supports three structurally different types of cones $\mathcal{K}$, which allows a surprisingly large number of nonlinear relations to be modelled (as described in the **MOSEK** modeling cookbook), while preserving the nice algorithmic and theoretical properties of linear optimization.

## 1.1 Why the Optimization Toolbox for MATLAB?

The Optimization Toolbox for MATLAB provides access to most of the functionality of **MOSEK** from a MATLAB environment. In addition the toolbox includes functions that replace functions from the MATLAB optimization toolbox available from MathWorks.

The Optimization Toolbox for MATLAB provides access to:

- Linear Optimization (LO)
- Conic Quadratic (Second-Order Cone) Optimization (CQO, SOCO)
- Convex Quadratic and Quadratically Constrained Optimization (QCQO)
- Semidefinite Optimization (SDO)
- Separable Convex Optimization (SCO)

as well as to additional functions for:

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

CHAPTER

TWO

# 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** | http://blog.mosek.com/ |
| **Google Group** | https://groups.google.com/forum/#!forum/mosek |
| **Twitter** | https://twitter.com/mosektw |
| **Google+** | https://plus.google.com/+Mosek/posts |
| **Linkedin** | https://www.linkedin.com/company/mosek-aps |

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

# CHAPTER THREE

# LICENSE AGREEMENT

Before using the **MOSEK** software, please read the license agreement available in the distribution at `<MSKHOME>/mosek/8/mosek-eula.pdf` or on the **MOSEK** website https://mosek.com/products/license-agreement.

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

## *zlib*

**MOSEK** includes 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** includes 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.
*
***************************************************************/
```

CHAPTER

# FOUR

# INSTALLATION

In this section we discuss how to install and setup the **MOSEK** Optimization Toolbox for MATLAB.

---

**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.

---

### Compatibility

The Optimization Toolbox for MATLAB can be used with MATLAB version r2014a or newer.

### Locating Files

The files in Optimization Toolbox for MATLAB are organized as reported in Table 4.1.

Table 4.1: Relevant files for the Optimization Toolbox for MATLAB.

| Relative Path | Description | Label |
|---|---|---|
| `<MSKHOME>/mosek/8/toolbox/r2014a` | Toolbox | `<TOOLBOXDIR>` |
| `<MSKHOME>/mosek/8/toolbox/r2014aom` | Toolbox (without overloading) | `<TOOLBOXOMDIR>` |
| `<MSKHOME>/mosek/8/toolbox/examples` | Examples | `<EXDIR>` |
| `<MSKHOME>/mosek/8/toolbox/data` | Additional data | `<MISCDIR>` |

where `<MSKHOME>` is the folder in which the **MOSEK** package has been installed.

### Setting up the paths

To use Optimization Toolbox for MATLAB the path to the toolbox directory must be added via the `addpath` command in MATLAB. Use the command

```
addpath <MSKHOME>/mosek/8/toolbox/r2014a
```

or, if you do not want to overload functions such as *linprog* and *quadprog* from the MATLAB Optimization Toolbox with their **MOSEK** versions, then write

```
addpath <MSKHOME>/mosek/8/toolbox/r2014aom
```

On the Windows platform the relevant paths are

```
addpath <MSKHOME>\mosek\8\toolbox\r2014a
addpath <MSKHOME>\mosek\8\toolbox\r2014aom
```

Alternatively, the path to Optimization Toolbox for MATLAB may be set from the command line or it can be added to MATLAB permanently using the configuration file `startup.m` or from the FileSet Path menu item. We refer to MATLAB documentation for details.

## 4.1 Testing the installation

You can verify that Optimization Toolbox for MATLAB works by executing

```
mosekdiag
```

in MATLAB. This should produce a message similar to this:

```
Matlab version: 8.3.0.532 (R2014a)
Architecture  : GLNXA64
Warning: The mosek optimizer could not be invoked from the command line. Most likely the path␣
↪has not been configured correctly. The mosek optimizer can still be invoked from the MATLAB␣
↪environment.
> In mosekdiag at 23
mosekopt: /home/andrea/mosek/8/toolbox/r2014a/mosekopt.mexa64

MOSEK Version 8.0.0.34(BETA) (Build date: 2016-8-16 00:52:47)
Copyright (c) MOSEK ApS, Denmark. WWW: mosek.com
Platform: Linux/64-X86

Found MOSEK version : major(8), minor(0), build(0), revision(34)
mosekopt is working correctly.
Warning: MOSEK Fusion is not configured correctly; check that mosek.jar is added to the␣
↪javaclasspath.
```

**Note:** If you only want to use Optimization Toolbox for MATLAB then the warnings about command line and *Fusion* interface can be ignored.

### 4.1.1 Troubleshooting

#### Missing library files such as libmosek64.8.1.dylib or similar

If you are using Mac OS and get an error such as

```
Library not loaded: libmosek64.8.1.dylib
Referenced from:
/Users/.../mosek/8/toolbox/r2014a/mosekopt.mexmaci64
Reason: image not found.

Error in callmosek>doCall (line 224)
[res,sol] = mosekopt('minimize info',prob,param);
```

then most likely you did not run the **MOSEK** installation script `install.py` found in the `bin` directory. See also the Installation guide for details.

#### Undefined Function or Variable *mosekopt*

If you get the MATLAB error message

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

you have not added the path to the Optimization Toolbox for MATLAB correctly as described above.

### Invalid MEX-file

For certain versions of Windows and MATLAB, the path to MEX files cannot contain spaces. Therefore, if you have installed **MOSEK** in `C:\Program Files\Mosek` and get a MATLAB error similar to:

```
Invalid MEX-file <MSKHOME>\Mosek\8\toolbox\r2014a\mosekopt.mexw64
```

try installing **MOSEK** in a different directory, for example `C:\Users\<someuser>\` .

### Output Arguments not assigned

If you encounter an error like

```
Error in ==> mosekpt at 1
function [r,res] = mosekopt(cmd,prob,param,callback)

Output argument "r" (and maybe others) not assigned during call to
"C:\Users\andrea\mosek\8\toolbox\r2014a\mosekopt.m>mosekopt".
```

then a mismatch between 32 and 64 bit versions of **MOSEK** and MATLAB is likely. From MATLAB type

```
which mosekopt
```

which (for a succesful installation) should point to a MEX file,

```
<MSKHOME>\mosek\8\toolbox\r2014a\mosekopt.mexw64
```

and not to a MATLAB `.m` file,

```
<MSKHOME>\mosek\8\toolbox\r2014a\mosekopt.m
```

CHAPTER

FIVE

# GUIDELINES

## 5.1 The MOSEK integration with MATLAB

In this section we provide some details concerning the integration of **MOSEK** with MATLAB. The information in this section is not strictly necessary for basic use of the **MOSEK** optimization toolbox for MATLAB.

### 5.1.1 The `mosekopt` MEX file

The central part of **MOSEK** optimization toolbox for MATLAB is the *mosekopt* MEX file. The mex file provides an interface to **MOSEK** that is employed by all the other **MOSEK** MATLAB functions. Therefore, we recommend to *mosekopt* function if possible because that give rise to the least overhead and provides the maximum of features.

### 5.1.2 Controlling log-output from `mosekopt`

Solver log-output is controlled using the `echo` parameter for *mosekopt*. The output is directed to the MATLAB console window. In newer versions of MATLAB (2015 or newer), the console output is not displayed until after the solver has terminated, which is a nuisance for long-running optimization tasks. As an accomodation, the log output can be copied to a log-file, using the `log` parameter for *mosekopt*. This log-file can then be inspected during the optimization task.

### 5.1.3 Compatibility with the MATLAB Optimization Toolbox

For compatibility with the MATLAB Optimization Toolbox, **MOSEK** provides the following functions:

- *linprog*: Solves linear optimization problems.
- *intlinprog*: Solves a linear optimization problem with integer constrained variables.
- *quadprog*: Solves quadratic optimization problems.
- *lsqlin*: Minimizes a least-squares objective with linear constraints.
- *lsqnonneg*: Minimizes a least-squares objective with nonnegativity constraints.
- *mskoptimget*: Getting an `options` structure for MATLAB compatible functions.
- *mskoptimset*: Setting up an `options` structure for MATLAB compatible functions.

These functions are described in detail in Sec. 17.1. The functions *mskoptimget* and *mskoptimset* are not fully compatible with the MATLAB counterparts, `optimget` and `optimset`, so the **MOSEK** versions should only be used in conjuctions with the **MOSEK** implementations of *linprog*, etc., and similarly `optimget` should be used in conjunction with the MATLAB implementations.

The corresponding MATLAB file for each function is located in the `toolbox/solvers` directory of the **MOSEK** distribution.

### 5.1.4 MOSEK and the MATLAB Parallel Computing Toolbox

Running **MOSEK** with the MATLAB Parallel Computing Toolbox requires multiple **MOSEK** licenses, since each thread runs a separate instance of the **MOSEK** optimizer. Each thread thus requires a **MOSEK** license.

## 5.2 Caveats Using the MATLAB Compiler

When using **MOSEK** with the MATLAB compiler it is necessary manually

- to remove `mosekopt.m` before compilation,
- copy the MEX file to the directory with MATLAB binary files and
- copy the `mosekopt.m` file back after compilation.

## 5.3 The license system

**MOSEK** is requires a license when used which is implemented as follows

1. a license token is checked out when any **MOSEK** function involving optimization, as for instance *mosekopt* is called the first time and
2. it is returned when MATLAB is terminated.

Now if the license should be checked in after use and hence be made available for another user then the license caching should be disabled as follows

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

Alternatively the command

```
mosekopt('nokeepenv')
```

will free all unused **MOSEK** licenses.

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

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

CHAPTER

SIX

# BASIC TUTORIALS

In this section a number of examples is provided to demonstrate the functionality required for solving linear, conic, semidefinite and quadratic problems as well as mixed integer problems.

- *Basic tutorial* : This is the simplest tutorial: it solves a linear optimization problem read from file. It will show how
  - setup the **MOSEK** environment and problem task,
  - run the solver and
  - check the optimization results.
- *Linear optimization tutorial* : It shows how to input a linear program. It will show how
  - define variables and their bounds,
  - define constraints and their bounds,
  - define a linear objective function,
  - input a linear program but rows or by column.
  - retrieve the solution.
- *Conic quadratic optimization tutorial* : The basic steps needed to formulate a conic quadratic program are introduced:
  - define quadratic cones,
  - assign the relevant variables to their cones.
- *Semidefinite optimization tutorial* : How to input semidefintite optimization problems is the topic of this tutorial, and in particular how to
  - input semidefinite matrices and in sparse format,
  - add semidefinite matrix variable and
  - formulate linear constraints and objective function based on matrix variables.
- *Mixed-Integer optimization tutorial* : This tutorial shows how integrality conditions can be specified.
- *Quadratic optimization tutorial* : It shows how to input quadratic terms in the objective function and constraints.
- *Response code tutorial* : How to deal with the termination and solver status code is the topic of this tutorial:
  - what are termination and termination code,
  - how to check for errors and
  - which are the best practice to deal with them.

  This is a very important tutorial, every user should go through it.
- *Reoptimization tutorial* : This tutorial gives information on how to

  - modify linear constraints,
  - add new variables/constraints and
  - reoptimize the given problem, i.e. run the **MOSEK** optimizer again.

- *Solution analysis* : This tutorial shows how the user can analyze the solution returned by the solver.
- *Parameter setting tutorial* : This tutorial shows how to set the solver parameters.

## 6.1 The Basics Tutorial

The simplest program using the **MOSEK** Matlab interface can be described shortly:

1. Load a problem into a problem structure (a *task*).
2. Optimize the problem.
3. Fetch the result.

Listing 6.1: A simple script that reads a problem from file and solves it.

```
%
%  Copyright : Copyright (c) MOSEK ApS, Denmark. All rights reserved.
%
%  File :      simple.m
%
%  Purpose :   To demonstrate how solve a problem
%              read from file.
%

function simple(inputfile, solfile)

cmd      = sprintf('read(%s)', inputfile)
% Read the problem from file
[rcode, res] = mosekopt(cmd)

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

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



end
```

## 6.2 Linear Optimization

The simplest optimization problem is a purely linear problem. A *linear optimization problem* is a problem of the following form:

Minimize or maximize the objective function

$$\sum_{j=0}^{n-1} c_j x_j + c^f$$

subject to the linear constraints

$$l_k^c \leq \sum_{j=0}^{n-1} a_{kj} x_j \leq u_k^c, \quad k = 0, \ldots, m-1,$$

and the bounds

$$l_j^x \leq x_j \leq u_j^x, \quad j = 0, \ldots, n-1.$$

The problem description consists of the following elements:

- $m$ and $n$ — the number of constraints and variables, respectively,
- $x$ — the variable vector of length $n$,
- $c$ — the coefficient vector of length $n$

$$c = \begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix},$$

- $c^f$ — fixed term in the objective,
- $A$ — an $m \times n$ matrix of coefficients

$$A = \begin{bmatrix} a_{0,0} & \cdots & a_{0,(n-1)} \\ \vdots & \cdots & \vdots \\ a_{(m-1),0} & \cdots & a_{(m-1),(n-1)} \end{bmatrix},$$

- $l^c$ and $u^c$ — the lower and upper bounds on constraints,
- $l^x$ and $u^x$ — the lower and upper bounds on variables.

Please note that we are using 0 as the first index: $x_0$ is the first element in variable vector $x$.

### 6.2.1 Example LO1

The following is an example of a small linear optimization problem:

$$\begin{array}{lccccccccc} \text{maximize} & 3x_0 & + & 1x_1 & + & 5x_2 & + & 1x_3 & & \\ \text{subject to} & 3x_0 & + & 1x_1 & + & 2x_2 & & & = & 30, \\ & 2x_0 & + & 1x_1 & + & 3x_2 & + & 1x_3 & \geq & 15, \\ & & & 2x_1 & & & + & 3x_3 & \leq & 25, \end{array} \tag{6.1}$$

under the bounds

$$\begin{array}{ccccc} 0 & \leq & x_0 & \leq & \infty, \\ 0 & \leq & x_1 & \leq & 10, \\ 0 & \leq & x_2 & \leq & \infty, \\ 0 & \leq & x_3 & \leq & \infty. \end{array}$$

#### Example: Linear optimization using *msklpopt*

A linear optimization problem such as (6.1) can be solved using the `msklpopt` function. The first step in solving the example (6.1) is to setup the data for problem (6.1) i.e. the $c$, $A$, etc. Afterwards the problem is solved using an appropriate call to `msklpopt`.

Listing 6.2: Script implementing problem (6.1).

```
function lo1()

c     = [3 1 5 1]';
a     = [[3 1 2 0];[2 1 3 1];[0 2 0 3]];
blc   = [30 15  -inf]';
buc   = [30 inf 25 ]';
blx   = zeros(4,1);
bux   = [inf 10 inf inf]';

[res] = msklpopt(c,a,blc,buc,blx,bux,[],'maximize');
sol   = res.sol;

% Interior-point solution.

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

% Basic solution.

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

Please note that

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

### Example: Linear optimization using *mosekopt*

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

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

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

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

The following MATLAB commands demonstrate how to set up the `prob` structure for the example (6.1) and solve the problem using *mosekopt*.

Listing 6.3: Script implementing problem (6.1) using *mosekopt*.

```
function lo2()
clear prob;

% Specify the c vector.
prob.c  = [3 1 5 1]';
```

```
% Specify a in sparse format.
subi   = [1 1 1 2 2 2 2 3 3];
subj   = [1 2 3 1 2 3 4 2 4];
valij  = [3 1 2 2 1 3 1 2 3];

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

% Specify lower bounds of the constraints.
prob.blc = [30 15  -inf]';

% Specify  upper bounds of the constraints.
prob.buc = [30 inf 25 ]';

% Specify lower bounds of the variables.
prob.blx = zeros(4,1);

% Specify upper bounds of the variables.
prob.bux = [inf 10 inf inf]';

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

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

Please note that

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

### Example: Linear optimization using *linprog*

**MOSEK** also provides a *linprog* function, which is compatible with the function provided by the MATLAB toolbox, using the syntax

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

Several control parameters can be set using the `options` structure, for example,

```
options.Write = 'test.opf';
linprog(f,A,b,B,c,l,u,x0,options);
```

creates a human readable `opf` file of the problem, and

```
options.Write = 'test.task';
linprog(f,A,b,B,c,l,u,x0,options);
```

creates a binary task file which can be send to **MOSEK** for debugging assistance or reporting errors.

Consult Sec. 5.1 for details on using *linprog* and other compatibility functions.

Internally, the *linprog* function is just a wrapper for the *mosekopt* function, and is mainly intended for compatibility reasons; advanced features are mainly available through the *mosekopt* function.

## 6.3 Conic Quadratic Optimization

Conic optimization is a generalization of linear optimization, allowing constraints of the type

$$x^t \in \mathcal{K}_t,$$

where $x^t$ is a subset of the problem variables and $\mathcal{K}_t$ is a convex cone. Since the set $\mathbb{R}^n$ of real numbers is also a convex cone, we can simply write a compound conic constraint $x \in \mathcal{K}$ where $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_l$ is a product of smaller cones and $x$ is the full problem variable.

**MOSEK** can solve conic quadratic optimization problems of the form

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

where the domain restriction, $x \in \mathcal{K}$, implies that all variables are partitioned into convex cones

$$x = (x^0, x^1, \ldots, x^{p-1}), \quad \text{with } x^t \in \mathcal{K}_t \subseteq \mathbb{R}^{n_t}.$$

For convenience, a user defining a conic quadratic problem only needs to specify subsets of variables $x^t$ belonging to quadratic cones. These are:

- Quadratic cone:

$$\mathcal{Q}^n = \left\{ x \in \mathbb{R}^n : x_0 \geq \sqrt{\sum_{j=1}^{n-1} x_j^2} \right\}.$$

- Rotated quadratic cone:

$$\mathcal{Q}_r^n = \left\{ x \in \mathbb{R}^n : 2x_0 x_1 \geq \sum_{j=2}^{n-1} x_j^2, \quad x_0 \geq 0, \quad x_1 \geq 0 \right\}.$$

For example, the following constraint:

$$(x_4, x_0, x_2) \in \mathcal{Q}^3$$

describes a convex cone in $\mathbb{R}^3$ given by the inequality:

$$x_4 \geq \sqrt{x_0^2 + x_2^2}.$$

Furthermore, each variable may belong to one cone at most. The constraint $x_i - x_j = 0$ would however allow $x_i$ and $x_j$ to belong to different cones with same effect.

### 6.3.1 Example CQO1

Consider the following conic quadratic problem which involves some linear constraints, a quadratic cone and a rotated quadratic cone.

$$\begin{array}{lccc} \text{minimize} & x_4 + x_5 + x_6 & & \\ \text{subject to} & x_1 + x_2 + 2x_3 & = & 1, \\ & x_1, x_2, x_3 & \geq & 0, \\ & x_4 \geq \sqrt{x_1^2 + x_2^2}, & & \\ & 2x_5 x_6 \geq x_3^2 & & \end{array} \tag{6.2}$$

The linear constraints are specified as if the problem was a linear problem whereas the cones are specified using two index lists `cones.subptr` and `cones.sub` and list of cone-type identifiers `cones.type`. The elements of all the cones are listed in `cones.sub`, and `cones.subptr` specifies the index of the first element in `cones.sub` for each cone.

Listing 6.4 demonstrates how to solve the example (6.2) using **MOSEK**.

Listing 6.4: Script implementing problem (6.2).

```
function cqo1()

clear prob;

[r, res] = mosekopt('symbcon');
% Specify the non-conic part of the problem.

prob.c   = [0 0 0 1 1 1];
prob.a   = sparse([1 1 2 0 0 0]);
prob.blc = 1;
prob.buc = 1;
prob.blx = [0 0 0 -inf -inf -inf];
prob.bux = inf*ones(6,1);

% Specify the cones.

prob.cones.type   = [res.symbcon.MSK_CT_QUAD, res.symbcon.MSK_CT_RQUAD];
prob.cones.sub    = [4, 1, 2, 5, 6, 3];
prob.cones.subptr = [1, 4];
% The field 'type' specifies the cone types, i.e., quadratic cone
% or rotated quadratic cone. The keys for the two cone types are MSK_CT_QUAD
% and MSK_CT_RQUAD, respectively.
%
% The fields 'sub' and 'subptr' specify the members of the cones,
% i.e., the above definitions imply that
%   x(4) >= sqrt(x(1)^2+x(2)^2) and 2 * x(5) * x(6) >= x(3)^2.

% Optimize the problem.

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

% Display the primal solution.

res.sol.itr.xx'
```

Note in particular that:

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

## 6.4 Semidefinite Optimization

Semidefinite optimization is a generalization of conic quadratic optimization, allowing the use of matrix variables belonging to the convex cone of positive semidefinite matrices

$$\mathcal{S}_+^r = \left\{ X \in \mathcal{S}^r : z^T X z \geq 0, \quad \forall z \in \mathbb{R}^r \right\},$$

where $\mathcal{S}^r$ is the set of $r \times r$ real-valued symmetric matrices.

**MOSEK** can solve semidefinite optimization problems of the form

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

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

$$\langle A, B\rangle := \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} A_{ij}B_{ij}.$$

### 6.4.1 Example SDO1

We consider the simple optimization problem with semidefinite and conic quadratic constraints:

$$\begin{array}{lll} \text{minimize} & \left\langle \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}, \overline{X} \right\rangle + x_0 & \\ \text{subject to} & \left\langle \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \overline{X} \right\rangle + x_0 & = \quad 1, \\ & \left\langle \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \overline{X} \right\rangle + x_1 + x_2 & = \quad 1/2, \\ & x_0 \geq \sqrt{{x_1}^2 + {x_2}^2}, & \overline{X} \succeq 0, \end{array} \tag{6.3}$$

The problem description contains a 3-dimensional symmetric semidefinite variable which can be written explicitly as:

$$\overline{X} = \begin{bmatrix} \overline{X}_{00} & \overline{X}_{10} & \overline{X}_{20} \\ \overline{X}_{10} & \overline{X}_{11} & \overline{X}_{21} \\ \overline{X}_{20} & \overline{X}_{21} & \overline{X}_{22} \end{bmatrix} \in \mathcal{S}_+^3,$$

and a conic quadratic variable $(x_0, x_1, x_2) \in \mathcal{Q}^3$. The objective is to minimize

$$2(\overline{X}_{00} + \overline{X}_{10} + \overline{X}_{11} + \overline{X}_{21} + \overline{X}_{22}) + x_0,$$

subject to the two linear constraints

$$\begin{array}{rcl} \overline{X}_{00} + \overline{X}_{11} + \overline{X}_{22} + x_0 & = & 1, \\ \overline{X}_{00} + \overline{X}_{11} + \overline{X}_{22} + 2(\overline{X}_{10} + \overline{X}_{20} + \overline{X}_{21}) + x_1 + x_2 & = & 1/2. \end{array}$$

Listing 6.5 demonstrates how to solve this problem using **MOSEK**.

Listing 6.5: Code implementing problem (6.3).

```
function sdo1()
[r, res] = mosekopt('symbcon');

prob.c         = [1, 0, 0];

prob.bardim    = [3];
prob.barc.subj = [1, 1, 1, 1, 1];
prob.barc.subk = [1, 2, 2, 3, 3];
prob.barc.subl = [1, 1, 2, 2, 3];
prob.barc.val  = [2.0, 1.0, 2.0, 1.0, 2.0];

prob.blc = [1, 0.5];
prob.buc = [1, 0.5];

% It is a good practice to provide the correct
% dimmension of A as the last two arguments
% because it facilitates better error checking.
prob.a         = sparse([1, 2, 2], [1, 2, 3], [1, 1, 1], 2, 3);
```

```
prob.bara.subi = [1, 1, 1, 2, 2, 2, 2, 2, 2];
prob.bara.subj = [1, 1, 1, 1, 1, 1, 1, 1, 1];
prob.bara.subk = [1, 2, 3, 1, 2, 3, 2, 3, 3];
prob.bara.subl = [1, 2, 3, 1, 1, 1, 2, 2, 3];
prob.bara.val  = [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0];

prob.cones.type   = [res.symbcon.MSK_CT_QUAD];
prob.cones.sub    = [1, 2, 3];
prob.cones.subptr = [1];

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

X = zeros(3);
X([1,2,3,5,6,9]) = res.sol.itr.barx;
X = X + tril(X,-1)';

x = res.sol.itr.xx;
```

The solution $x$ is returned in `res.sol.itr.xx` and the numerical values of $\overline{X}_j$ are returned in `res.sol.barx`; the lower triangular part of each $\overline{X}_j$ is stacked column-by-column into an array, and each array is then concatenated forming a single array `res.sol.itr.barx` representing $\overline{X}_1, \ldots, \overline{X}_p$. Similarly, the dual semidefinite variables $\overline{S}_j$ are recovered through `res.sol.itr.bars`.

## 6.5 Quadratic Optimization

**MOSEK** can solve quadratic and quadratically constrained problems, as long as they are convex. This class of problems can be formulated as follows:

$$\begin{array}{lccccl} \mbox{minimize} & & & \frac{1}{2}x^TQ^ox + c^Tx + c^f & & \\ \mbox{subject to} & l_k^c & \leq & \frac{1}{2}x^TQ^kx + \sum_{j=0}^{n-1} a_{k,j}x_j & \leq & u_k^c, \quad k = 0, \ldots, m-1, \\ & l_j^x & \leq & x_j & \leq & u_j^x, \quad j = 0, \ldots, n-1. \end{array} \tag{6.4}$$

Without loss of generality it is assumed that $Q^o$ and $Q^k$ are all symmetric because

$$x^TQx = \frac{1}{2}x^T(Q + Q^T)x.$$

This implies that a non-symmetric $Q$ can be replaced by the symmetric matrix $\frac{1}{2}(Q + Q^T)$.

The problem is required to be convex. More precisely, the matrix $Q^o$ must be positive semi-definite and the $k$th constraint must be of the form

$$l_k^c \leq \frac{1}{2}x^TQ^kx + \sum_{j=0}^{n-1} a_{k,j}x_j \tag{6.5}$$

with a negative semi-definite $Q^k$ or of the form

$$\frac{1}{2}x^TQ^kx + \sum_{j=0}^{n-1} a_{k,j}x_j \leq u_k^c.$$

with a positive semi-definite $Q^k$. This implies that quadratic equalities are *not* allowed. Specifying a non-convex problem will result in an error when the optimizer is called.

A matrix is positive semidefinite if all the eigenvalues of $Q$ are nonnegative. An alternative statement of the positive semidefinite requirement is

$$x^TQx \geq 0, \quad \forall x.$$

If the convexity (i.e. semidefiniteness) conditions are not met **MOSEK** will not produce reliable results or work at all.

### 6.5.1 Example: Quadratic Objective

We look at a small problem with linear constraints and quadratic objective:

$$\begin{array}{lrl} \text{minimize} & & x_1^2 + 0.1x_2^2 + x_3^2 - x_1x_3 - x_2 \\ \text{subject to} & 1 \leq & x_1 + x_2 + x_3 \\ & 0 \leq & x. \end{array} \tag{6.6}$$

The matrix formulation (6.6) has:

$$Q^o = \begin{bmatrix} 2 & 0 & -1 \\ 0 & 0.2 & 0 \\ -1 & 0 & 2 \end{bmatrix}, c = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}, A = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix},$$

with the bounds:

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

Please note the explicit $\frac{1}{2}$ in the objective function of (6.4) which implies that diagonal elements must be doubled in $Q$, i.e. $Q_{11} = 2$, whereas the coefficient in (6.6) is 1 in front of $x_1^2$.

#### Using `mosekopt`

In Listing 6.6 we show how to use *mosekopt* to solve problem (6.6). This is the preferred way.

Listing 6.6: How to solve problem (6.6) using *mosekopt*.

```
function qo2()

clear prob;

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

% Define the data.

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

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

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

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

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

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

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

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

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

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

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

% Display primal solution for the variables.
res.sol.itr.xx'
```

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

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

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

### Using mskqpopt

In Listing 6.7 we show how to use *mskqpopt* to solve problem (6.6).

Listing 6.7: Function solving problem (6.6) using *mskqpopt*.

```
function qo1()

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

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

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

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

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

## 6.5.2 Example: Quadratic constraints

In this section we show how to solve a problem with quadratic constraints. Please note that quadratic constraints are subject to the convexity requirement (6.5).

Consider the problem:

$$\begin{array}{lrcl} \text{minimize} & & & x_1^2 + 0.1x_2^2 + x_3^2 - x_1x_3 - x_2 \\ \text{subject to} & 1 & \leq & x_1 + x_2 + x_3 - x_1^2 - x_2^2 - 0.1x_3^2 + 0.2x_1x_3, \\ & & & x \geq 0. \end{array}$$

This is equivalent to

$$\begin{array}{lccc} \text{minimize} & \frac{1}{2}x^TQ^ox + c^Tx & & \\ \text{subject to} & \frac{1}{2}x^TQ^0x + Ax & \geq & b, \\ & x \geq 0, & & \end{array} \tag{6.7}$$

where

$$Q^o = \left[\begin{array}{ccc} 2 & 0 & -1 \\ 0 & 0.2 & 0 \\ -1 & 0 & 2 \end{array}\right], c = \left[\begin{array}{ccc} 0 & -1 & 0 \end{array}\right]^T, A = \left[\begin{array}{ccc} 1 & 1 & 1 \end{array}\right], b = 1.$$

$$Q^0 = \left[\begin{array}{ccc} -2 & 0 & 0.2 \\ 0 & -2 & 0 \\ 0.2 & 0 & -0.2 \end{array}\right].$$

The linear parts and quadratic objective are set up the way described in the previous tutorial.

### Setting up quadratic constraints

Please note that there are quadratic terms in both constraints. This problem can be solved using *mosekopt* as the following

Listing 6.8: Script implementing problem (6.7).

```
function qcqo1()
clear prob;

% Specify the linear objective terms.
prob.c      = [0, -1, 0];

% Specify the quadratic terms of the constraints.
prob.qcsubk = [1     1     1    1  ]';
prob.qcsubi = [1     2     3    3  ]';
prob.qcsubj = [1     2     3    1  ]';
prob.qcval  = [-2.0 -2.0 -0.2 0.2]';

% Specify the quadratic terms of the objective.
prob.qosubi = [1     2     3     3  ]';
prob.qosubj = [1     2     3     1  ]';
prob.qoval  = [2.0   0.2   2.0 -1.0]';

% Specify the linear constraint matrix
prob.a      = [1 1 1];

% Specify the lower bounds
prob.blc    = [1];
prob.blx    = zeros(3,1);

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

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

## 6.6 Integer Optimization

An optimization problem where one or more of the variables are constrained to integer values is called a (mixed) integer optimization problem. **MOSEK** supports integer variables in combination with linear and conic quadratic problems. See the previous tutorials for an introduction to how to model these types of problems.

### 6.6.1 Example MILO1

We use the example

$$\begin{array}{lccl} \text{maximize} & x_0 + 0.64x_1 & & \\ \text{subject to} & 50x_0 + 31x_1 & \leq & 250, \\ & 3x_0 - 2x_1 & \geq & -4, \\ & x_0, x_1 \geq 0 & & \text{and integer} \end{array} \tag{6.8}$$

to demonstrate how to set up and solve a problem with integer variables. It has the structure of a linear optimization problem (see Sec. 6.2) except for integrality constraints on the variables. Therefore, only the specification of the integer constraints requires something new compared to the linear optimization problem discussed previously.

The complete source for the example is listed in Listing 6.9.

Listing 6.9: How to solve problem (6.8).

```
function milo1()
clear prob
prob.c        = [1 0.64];
prob.a        = [[50 31];[3 -2]];
prob.blc      = [-inf -4];
prob.buc      = [250 inf];
prob.blx      = [0 0];
prob.bux      = [inf inf];

% Specify indexes of variables that are integer
% constrained.

prob.ints.sub = [1 2];

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

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

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

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

**MOSEK** also provides a wrapper for the *intlinprog* function found in the MATLAB optimization toolbox. This function solves linear problems wth integer variables; see the reference section for details.

### 6.6.2 Specifying an initial solution

Solution time of can often be reduced by providing an initial solution for the solver. It is not necessary to specify the whole solution. By setting the *MSK_IPAR_MIO_CONSTRUCT_SOL* parameter to *"MSK_ON"* and inputting values for the integer variables only, **MOSEK** will be forced to compute the remaining continuous variable values. If the specified integer solution is infeasible or incomplete, **MOSEK** will simply ignore it.

We concentrate on a simple example below.

$$
\begin{array}{ll}
\text{maximize} & 7x_0 + 10x_1 + x_2 + 5x_3 \\
\text{subject to} & x_0 + x_1 + x_2 + x_3 \leq 2.5 \\
& x_0, x_1, x_2 \in \mathbb{Z} \\
& x_0, x_1, x_2, x_3 \geq 0
\end{array} \tag{6.9}
$$

Listing 6.10: Script solving problem (6.9).

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

clear prob

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

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

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

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

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

## 6.7 Optimizer Termination Handling

After solving an optimization problem with **MOSEK** an approriate action must be taken depending on the outcome. Usually the expected outcome is an optimal solution, but there may be several situations where this is not the result. E.g., if the problem is infeasible or nearly so or if the solver ran out of memory or stalled while optimizing, the result may not be as expected.

This section discusses what should be considered when an optimization has ended unsuccessfully.

Before continuing, let us consider the four status codes available in **MOSEK** that is relevant for the error handing:

- **Termination code**: It provides information about why the optimizer terminated. For instance if a time limit has been specfied (this is common for mixed integer problems), the termination code will tell if this termination limit was the cause of the termination. Note that reaching a prespecfied time limit is not considered an exceptional case. It must be expected that this occurs occasionally.
- **Response code**: It is an information about the system status and the outcome of the call to a **MOSEK** functionalities. This code is used to report the unexpected failures such as out of space.

The response code is the returned value of most functions of the API, and its type is *`rescode`*. See Sec. 17.5 for a list of possible return codes.

- **Solution status**: It contains information about the status of the solution, e.g., whether the solution is optimal or a certificate of infeasibility.
- **Problem status**: It describes what **MOSEK** knows about the feasibility of the problem, i.e., if the is problem feasible or infeasible.

The problem status is mostly used for integer problems. For continuous problems a problem status of, say, *infeasible* will always mean that the solution is a certificate of infeasibility. For integer problems it is not possible to provide a certificate, and thus a separate problem status is useful.

Note that if we want to report, e.g., that the optimizer terminated due to a time limit or because it stalled but with a feasible solution, we have to consider *both* the termination code, *and* the solution status.

The following pseudo code demonstrates a best practice way of dealing with the status codes.

- `if ( the solution status is as expected )`
  - **The normal case**:

    Do whatever that was planned. Note the response code is ignored because the solution has the expected status. Of course we may check the response anyway if we like.
- `else`
  - **Exceptional case**:

    Based on solution status, response and termination codes take appropriate action.

In Listing 6.11 the pseudo code is implemented. The idea of the example is to read an optimization problem from a file, e.g., an MPS file and optimize it. Based on status codes an appropriate action is taken, which in this case is to print a suitable message.

Listing 6.11: A typical code that handle **MOSEK** response code.

```
function response(inputfile, solfile)


  cmd      = sprintf('read(%s)', inputfile)
  % Read the problem from file
  [r, res] = mosekopt(cmd)

  if strcmp( res.rcodestr , 'MSK_RES_OK')

      % Perform the optimization.
      [r,res] = mosekopt('minimize', res.prob);
      r
      res
      %Expected result: The solution status of the basic solution is optimal.
      if strcmp(res.rcodestr, 'MSK_RES_OK')

          solsta = strcat('MSK_SOL_STA_', res.sol.itr.solsta)

          if strcmp( solsta , 'MSK_SOL_STA_OPTIMAL') || ...
             strcmp( solsta , 'MSK_SOL_STA_NEAR_OPTIMAL')
```

```
            fprintf('An optimal basic solution is located.');

        elseif strcmp( solsta , 'MSK_SOL_STA_DUAL_INFEAS_CER') || ...
               strcmp( solsta , 'MSK_SOL_STA_NEAR_DUAL_INFEAS_CER')
            fprintf('Dual infeasibility certificate found.');

        elseif strcmp( solsta , 'MSK_SOL_STA_PRIM_INFEAS_CER') || ...
               strcmp( soslta , 'MSK_SOL_STA_NEAR_PRIM_INFEAS_CER')
          fprintf('Primal infeasibility certificate found.');

        elseif strcmp( solsta , 'MSK_SOL_STA_UNKNOWN')

            % The solutions status is unknown. The termination code
            % indicates why the optimizer terminated prematurely.

            fprintf('The solution status is unknown.');

            if ~strcmp(res.rcodestr, 'MSK_RES_OK' )

                % A system failure e.g. out of space.
                fprintf('  Response code: %s\n', res);

            else

                %No system failure e.g. an iteration limit is reached.
                printf('  Termination code: %s\n', res);
            end

        else
          fprintf('An unexpected solution status is obtained.');
        end


    else
      fprintf('Could not obtain the solution status for the requested solution.');

  end

  fprintf('Return code: %d (0 means no error occurred.)\n',r);

end
```

## 6.8 Problem Modification and Reoptimization

Often one might want to solve not just a single optimization problem, but a sequence of problems, each differing only slightly from the previous one. This section demonstrates how to modify and re-optimize an existing problem. The example we study is a simple production planning model.

Problem modifications regarding variables, cones, objective function and constraints can be grouped in categories:

- add/remove,
- coefficient modifications,
- bounds modifications.

Especially removing variables and constraints can be costly. Special care must be taken with respect to constraints and variable indexes that may be invalidated.

Depending on the type of modification, **MOSEK** may be able to optimize the modified problem more efficiently exploiting the information and internal state from the previous execution. After optimization, the solution is always stored internally, and is available before next optimization. The former optimal solution may be still feasible, but no longer optimal; or it may remain optimal if the modification of the objective function was small. This special case is discussed in Sec. 15.

In general, **MOSEK** exploits dual information and availability of an optimal basis from the previous execution. The simplex optimizer is well suited for exploiting an existing primal or dual feasible solution. Restarting capabilities for interior-point methods are still not as reliable and effective as those for the simplex algorithm. More information can be found in Chapter 10 of the book *[Chv83]*.

### 6.8.1 Example: Production Planning

A company manufactures three types of products. Suppose the stages of manufacturing can be split into three parts: Assembly, Polishing and Packing. In the table below we show the time required for each stage as well as the profit associated with each product.

| Product no. | Assembly (minutes) | Polishing (minutes) | Packing (minutes) | Profit ($) |
|---|---|---|---|---|
| 0 | 2 | 3 | 2 | 1.50 |
| 1 | 4 | 2 | 3 | 2.50 |
| 2 | 3 | 3 | 2 | 3.00 |

With the current resources available, the company has $100,000$ minutes of assembly time, $50,000$ minutes of polishing time and $60,000$ minutes of packing time available per year. We want to know how many items of each product the company should produce each year in order to maximize profit?

Denoting the number of items of each type by $x_0, x_1$ and $x_2$, this problem can be formulated as a linear optimization problem:

$$\begin{array}{lccccccc} \text{maximize} & 1.5x_0 & + & 2.5x_1 & + & 3.0x_2 & & \\ \text{subject to} & 2x_0 & + & 4x_1 & + & 3x_2 & \leq & 100000, \\ & 3x_0 & + & 2x_1 & + & 3x_2 & \leq & 50000, \\ & 2x_0 & + & 3x_1 & + & 2x_2 & \leq & 60000, \end{array} \tag{6.10}$$

and

$$x_0, x_1, x_2 \geq 0.$$

Code in Listing 6.12 loads and solves this problem.

Listing 6.12: Setting up and solving problem (6.10)

```
% Specify the c vector.
prob.c  = [1.5 2.5 3.0]';

% Specify a in sparse format.
subi   = [1 1 1 2 2 2 3 3 3];
subj   = [1 2 3 1 2 3 1 2 3];
valij  = [2 4 3 3 2 3 2 3 2];

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

% Specify lower bounds of the constraints.
prob.blc = [-inf -inf -inf]';

% Specify  upper bounds of the constraints.
prob.buc = [100000 50000 60000]';

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

% Specify upper bounds of the variables.
```

```
prob.bux = [inf inf inf]';

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

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

### 6.8.2 Changing the Linear Constraint Matrix

Suppose we want to change the time required for assembly of product 0 to 3 minutes. This corresponds to setting $a_{0,0} = 3$, which is done by directly modifying the `A` matrix of the problem, as shown below.

```
prob.a(1,1) = 3.0
```

The problem now has the form:

$$\begin{array}{lccccccc} \text{maximize} & 1.5x_0 & + & 2.5x_1 & + & 3.0x_2 & & \\ \text{subject to} & 3x_0 & + & 4x_1 & + & 3x_2 & \leq & 100000, \\ & 3x_0 & + & 2x_1 & + & 3x_2 & \leq & 50000, \\ & 2x_0 & + & 3x_1 & + & 2x_2 & \leq & 60000, \end{array} \tag{6.11}$$

and

$$x_0, x_1, x_2 \geq 0.$$

After this operation we can reoptimize the problem.

### 6.8.3 Appending Variables

We now want to add a new product with the following data:

| Product no. | Assembly (minutes) | Polishing (minutes) | Packing (minutes) | Profit ($) |
|---|---|---|---|---|
| 3 | 4 | 0 | 1 | 1.00 |

This corresponds to creating a new variable $x_3$, appending a new column to the $A$ matrix and setting a new term in the objective. We do this in Listing 6.13

Listing 6.13: How to add a new variable (column)

```
prob.c      = [prob.c;1.0];
prob.a      = [prob.a,sparse([4.0 0. 1.0]')];
prob.blx    = zeros(4,1);
prob.bux    = [prob.bux; inf]
```

After this operation the new problem is:

$$\begin{array}{lccccccccc} \text{maximize} & 1.5x_0 & + & 2.5x_1 & + & 3.0x_2 & + & 1.0x_3 & & \\ \text{subject to} & 3x_0 & + & 4x_1 & + & 3x_2 & + & 4x_3 & \leq & 100000, \\ & 3x_0 & + & 2x_1 & + & 3x_2 & & & \leq & 50000, \\ & 2x_0 & + & 3x_1 & + & 2x_2 & + & 1x_3 & \leq & 60000, \end{array} \tag{6.12}$$

and

$$x_0, x_1, x_2, x_3 \geq 0.$$

### 6.8.4 Appending Constraints

Now suppose we want to add a new stage to the production process called *Quality control* for which 30000 minutes are available. The time requirement for this stage is shown below:

| Product no. | Quality control (minutes) |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 1 |
| 3 | 1 |

This corresponds to adding the constraint

$$x_0 + 2x_1 + x_2 + x_3 \leq 30000$$

to the problem. This is done as follows.

Listing 6.14: Adding a new constraint.

```
prob.a      = [prob.a;sparse([1.0 2.0 1.0 1.0])];
prob.blc    = [prob.blc;30000.0];
prob.buc    = [prob.buc;-inf];
```

Again, we can continue with re-optimizing the modified problem.

## 6.9 Solution Analysis

The main purpose of **MOSEK** is to solve optimization problems and therefore the most fundamental question to be asked is whether the solution reported by **MOSEK** is a solution to the desired optimization problem.

There can be several reasons why it might be not case. The most prominent reasons are:

- A wrong problem. The problem inputted to **MOSEK** is simply not the right problem, i.e. some of the data may have been corrupted or the model has been incorrectly built.
- Numerical issues. The problem is badly scaled or otherwise badly posed.
- Other reasons. E.g. not enough memory or an explicit user request to stop.

The first step in verifying that **MOSEK** reports the expected solution is to inspect the solution summary generated by **MOSEK** (see Sec. 6.9.1). The solution summary provides information about

- the problem and solution statuses,
- objective value and infeasibility measures for the primal solution, and
- objective value and infeasibility measures for the dual solution, where applicable.

By inspecting the solution summary it can be verified that **MOSEK** produces a feasible solution, and, in the continuous case, the optimality can be checked using the dual solution. Furthermore, the problem itself ca be inspected using the problem analyzer discussed in Sec. 13.

If the summary reports conflicting information (e.g. a solution status that does not match the actual solution), or the cause for terminating the solver before a solution was found cannot be traced back to the reasons stated above, it may be caused by a bug in the solver; in this case, please contact **MOSEK** support (see Sec. 2).

If it has been verified that **MOSEK** solves the problem correctly but the solution is still not as expected, next step is to verify that the primal solution satisfies all the constraints. Hence, using the original problem it must be determined whether the solution satisfies all the required constraints in the model. For instance assume that the problem has the constraints

$$\begin{array}{rcl} x_1 + 2x_2 + x_3 & \leq & 1, \\ x_1, x_2, x_3 \geq 0 & & \end{array}$$

and **MOSEK** reports the optimal solution

$$x_1 = x_2 = x_3 = 1.$$

Then clearly the solution violates the constraints. The most likely explanation is that the model does not match the problem entered into **MOSEK**, for instance

$$x_1 - 2x_2 + x_3 \leq 1$$

may have been inputted instead of

$$x_1 + 2x_2 + x_3 \leq 1.$$

A good way to debug such an issue is to dump the problem to *OPF file* and check whether the violated constraint has been specified correctly.

Verifying that a feasible solution is optimal can be harder. However, for continuous problems, i.e. problems without any integer constraints, optimality can verified using a dual solution. Normally, **MOSEK** will report a dual solution; if that is feasible and has the same objective value as the primal solution, then the primal solution must be optimal.

An alternative method is to find another primal solution that has better objective value than the one reported to **MOSEK**. If that is possible then either the problem is badly posed or there is bug in **MOSEK**.

### 6.9.1 The Solution Summary

Due to **MOSEK** employs finite precision floating point numbers then reported solution is an approximate optimal solution. Therefore after solving an optimization problem it is relevant to investigate how good an approximation the solution is. For a convex optimization problem that is an easy task because the optimality conditions are:

- The primal solution must satisfy all the primal constraints.
- The dual solution much satisfy all the dual constraints.
- The primal and dual objective values must be identical.

Therefore, the **MOSEK** solution summary displays that information that makes it possible to verify the optimality conditions. Indeed the solution summary reports how much primal and dual solutions violate the primal and constraints respectively. In addition the objective values assoctaied with each solution repoted.

In case of a linear optimization problem the solution summary may look like

```
Basic solution summary
  Problem status  : PRIMAL_AND_DUAL_FEASIBLE
  Solution status : OPTIMAL
  Primal.  obj: -4.6475314286e+002  nrm: 5e+002   Viol.  con: 1e-014   var: 1e-014
  Dual.    obj: -4.6475314543e+002  nrm: 1e+001   Viol.  con: 4e-009   var: 4e-016
```

The interpreation of the solution summary is as follows:

- Information for the basic solution is reported.
- The problem status is primal and dual feasible which means the problem has an optimal solution.
- The solution status is optimal.

- Next information about the primal solution is reported. The information consists of the objective value, the infinity norm of the primal solution and violation meassures. The violation for the constraints (`con:`) is the maximal violation in any of the constraints. Whereas the violations for the variables (`var:`) is the maximal bound violation for any of the variables. In this case the primal violations for the constraints and variables are small meaning the solution is an almost feasible solution. Observe due to the rounding errors it can be expected that the violations are proportional to the size (`nrm:`) of the solution.
- Similarly for the dual solution the violations are small and hence the dual solution is almost feasible.
- Finally, it can be seen that the primal and dual objective values are almost identical.

To summarize in this case a primal and a dual solution only violate the primal and dual constraints slightly. Moreover, the primal and dual objective values are almost identical and hence it can be concluded that the reported solution is a good approximation to the optimal solution.

The reason the size (=norms) of the solution are shown is that it shows some about conditioning of the problem because if the primal and/or dual solution has very large norm then the violations and objective values are sensitive to small pertubations in the problem data. Therefore, the problem is unstable and care should be taken before using the solution.

Now what happens if the problem does not have an optimal solution e.g. is primal infeasible. In such a case the solution summary may look like

```
Interior-point solution summary
  Problem status  : PRIMAL_INFEASIBLE
  Solution status : PRIMAL_INFEASIBLE_CER
  Dual.    obj: 6.7319732555e+000   nrm: 8e+000   Viol.  con: 3e-010   var: 2e-009
```

i.e. **MOSEK** reports that the solution is a certificate of primal infeasibility but a certificate of primal infeasibility what does that mean? It means that the dual solution is a Farkas type certificate. Recall Farkas' Lemma says

$$\begin{array}{rcl} Ax & = & b, \\ x & \geq & 0 \end{array}$$

if and only if a $y$ exists such that

$$\begin{array}{rcl} A^T y & \leq & 0, \\ b^T y & > & 0. \end{array} \tag{6.13}$$

Observe the infeasibility certificate has the same form as a regular dual solution and therefore the certificate is stored as a dual solution. In order to check quality of the primal infeasibility certificate it should be checked whether satisfies (6.13). Hence, the dual objective value is $b^T y$ should be strictly positive and the maximal violation in $A^T y \leq 0$ should be a small. In this case we conclude the certificate is of high quality because the dual objective is postive and large compared to the violations. Note the Farkas certificate is a ray so any postive multiple of that ray is also certificate. This implies the absolute of the value objective value and the violation is not relevant.

In the case a problem is dual infeasible then the solution summary may look like

```
Basic solution summary
Problem status  : DUAL_INFEASIBLE
Solution status : DUAL_INFEASIBLE_CER
Primal.  obj: -2.0000000000e-002  nrm: 1e+000   Viol.  con: 0e+000   var: 0e+000
```

Observe when a solution is a certificate of dual infeasibility then the primal solution contains the certificate. Moreoever, given the problem is a minimization problem the objective value should be negative and large compared to the worst violation if the certificate is strong.

Listing 6.15 shows how to use these function to determine the quality of the solution.

Listing 6.15: An example of solution quality analysis.

```
function solutionquality(data)

    cmd      = sprintf('read(%s)',data)
    % Read the problem from file
    [r, res] = mosekopt(cmd)

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

    solsta = strcat('MSK_SOL_STA_', res.sol.itr.solsta);

    if strcmp(solsta, 'MSK_SOL_STA_OPTIMAL') || strcmp(solsta, 'MSK_SOL_STA_NEAR_OPTIMAL')

      sol = res.sol.itr
      primalobj= sol.pobjval
      dualobj= sol.dobjval

      abs_obj_gap     = abs(dualobj - primalobj);
      rel_obj_gap     = abs_obj_gap/(1.0 + min( abs(primalobj), abs(dualobj)));


      % Assume the application needs the solution to be within
      % 1e-6 optimality in an absolute sense. Another approach
      % would be looking at the relative objective gap */

      fprintf('\n\n');
      fprintf('Customized solution information.\n');
      fprintf('  Absolute objective gap: %e\n',abs_obj_gap);
      fprintf('  Relative objective gap: %e\n',rel_obj_gap);

      accepted = 1;

      if ( rel_obj_gap>1e-6 )
          fprintf('Warning: The relative objective gap is LARGE.\n');
          accepted = 0;
      end

      if ( accepted )
          res.sol.itr.xx
      else
          % Print detailed information about the solution
          r = MSK_analyzesolution(task,MSK_STREAM_LOG,whichsol);
      end

    elseif strcmp(solsta, 'MSK_SOL_STA_DUAL_INFEAS_CER') || ...
          strcmp(solsta, 'MSK_SOL_STA_PRIM_INFEAS_CER') || ...
          strcmp(solsta, 'MSK_SOL_STA_NEAR_DUAL_INFEAS_CER') || ...
          strcmp(solsta, 'MSK_SOL_STA_NEAR_PRIM_INFEAS_CER')
        fprintf('Primal or dual infeasibility certificate found.\n');

    elseif strcmp(solsta,  'MSK_SOL_STA_UNKNOWN')
        fprintf('The status of the solution is unknown.\n');

    else
        fprintf('Other solution status');
    end

end
```

### 6.9.2 The Solution Summary for Mixed-Integer Problems

The solution summary for a mixed-integer problem may look like

Listing 6.16: Example of solution summary for a mixed-integer problem.

```
Integer solution solution summary
Problem status  : PRIMAL_FEASIBLE
Solution status : INTEGER_OPTIMAL
Primal.  obj: 3.4016000000e+005   nrm: 1e+000   Viol.  con: 0e+000   var: 0e+000   itg: 3e-014
```

The main diffrence compared to thecontinous case covered previously is that no information about the dual solution is provided. Simply because there is no dual solution available for a mixed integer problem. In this case it can be seen that the solution is highly feasible because the violations are small. Moreoever, the solution is denoted integer optimal. Observe *itg: 3e-014* implies that all the integer constrained variables are at most $3e - 014$ from being an exact integer.

## 6.10 Solver Parameters

The **MOSEK** API provides many parameters to tune and customize the solver behaviour. Parameters are grouped depending on their type: integer, double or string. In general, it should not be necessary to change any of the parameters but if required, it is easily done. A complete list of all parameters is found in Sec. 17.4.

We will show how to access and set the integer parameter that define the logging verbosity of the solver, i.e. *MSK_IPAR_LOG*, and the algorithm used by **MOSEK**, i.e. *MSK_IPAR_OPTIMIZER*.

---

**Note:** The very same concepts and procedures apply to string and double valued parameters.

---

To inspect the current value of a parameter, we can use the *mosekopt* command `param`:

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

To set a parameter we only need to make a structure with fields that corresponds to the parameters we want to set:

```
param.MSK_IPAR_LOG = 1
```

```
param.MSK_IPAR_LOG = -1
```

The values for integer parameters are either simple integer values or enum values. Enumerations are provided mainly to improve readability and ensure compatibility.

In the next lines we show how to set the algorithm used by **MOSEK** to solve linear optimization problem. To that purpose we set the *MSK_IPAR_OPTIMIZER* parameter using a value from the *optimizertype* enumeration: for instance we may decide to use the dual simplex algorithm, and thus

```
param.MSK_IPAR_OPTIMIZER = 'MSK_OPTIMIZER_DUAL_SIMPLEX'
```

For more information about other parameter related functions, please browse the API reference in Sec. 17.1.

The complete code for this tutorial follows in Listing 6.17.

Listing 6.17: Parameter setting example.

```
function r = parameters()
```

```
fprintf('Test MOSEK parameter get/set functions');

[r,resp]=mosekopt('param');

fprintf('Default value for parameter MSK_IPAR_LOG= %d\n', resp.param.MSK_IPAR_LOG)

fprintf(' setting to 1...');
param.MSK_IPAR_LOG = 1


fprintf(' setting to -1 ...');
param.MSK_IPAR_LOG = -1


fprintf(' selecting the dual simplex algorithm...');
param.MSK_IPAR_OPTIMIZER = 'MSK_OPTIMIZER_DUAL_SIMPLEX'

try
    % Perform the optimization, but it should fail
    [r,resp] = mosekopt('minimize', [] , param);
catch
    fprintf('The value -1 for parameter MSK_IPAR_LOG has been correctly detected as wrong!')
    r = 0
    return
end

fprintf('The value -1 for parameter MSK_IPAR_LOG has NOT been correctly detected as wrong!')

r = 1

end
```

CHAPTER

SEVEN

# NONLINEAR TUTORIALS

This chapter provides information about how to solve general convex nonlinear optimization problems using **MOSEK**. By general nonlinear problems we mean those that cannot be formulated in conic or convex quadratically constrained form.

In general we recommend not to use the general nonlinear optimizer unless absolutely necessary. The reasons are:

- The algorithm employed for nonlinear optimization problems is not as efficient as the one employed for conic problems. Conic problems have special structure that can be exploited to make the optimizer faster and more robust.
- **MOSEK** has no way of checking whether the formulated problem is convex and if this assumption is not satisfied the optimizer will not work.
- The nonlinear optimizer requires 1st and 2nd order derivative information which is often hard to provide correctly.

Instead, we advise:

- Consider reformulating the problem to a conic quadratic optimization problem if at all possible. In particular many problems involving polynomial terms can easily be reformulated to conic quadratic form.
- Consider reformulating the problem to a separable optimization problem because that simplifies the issue with verifying convexity and computing 1st and 2nd order derivatives significantly. In most cases problems in separable form also solve faster because of the simpler structure of the functions.
- Finally, if the problem cannot be reformulated in separable form use a modelling language like AMPL or GAMS, which will perform all the preprocessing, computing function values and derivatives. This eliminates an important source of errors. Therefore, it is strongly recommended to use a modelling language at the prototype stage.

The Optimization Toolbox for MATLAB provides the following nonlinear interfaces:

## 7.1 Separable Convex (SCopt) Interface

The Optimization Toolbox for MATLAB provides a way to add simple non-linear functions composed from a limited set of non-linear terms. Non-linear terms can be mixed with quadratic terms in objective and constraints. We consider problems which can be formulated as:

$$\begin{array}{lccccl} \text{minimize} & & & z_0(x) + c^T x & & \\ \text{subject to} & l_i^c & \leq & z_i(x) + a_i^T x & \leq & u_i^c \quad i = 1 \ldots m \\ & l^x & \leq & x & \leq & u^x, \end{array}$$

where $x \in \mathbb{R}^n$ and each $z_i : \mathbb{R}^n \to \mathbb{R}$ is separable, that is can be written as a sum

$$z_i(x) = \sum_{j=1}^{n} z_{i,j}(x_j).$$

The interface implements a limited set of functions which can appear as $z_{i,j}$. They are:

Table 7.1: Functions supported by the SCopt interface.

| Separable function | Operator name | Name |
|---|---|---|
| $fx\ln(x)$ | *ent* | Entropy function |
| $fe^{gx+h}$ | *exp* | Exponential function |
| $f\ln(gx+h)$ | *log* | Logarithm |
| $f(x+h)^g$ | *pow* | Power function |

where $f, g, h \in \mathbb{R}$ are constants. This formulation does not guarantee convexity. For **MOSEK** to be able to solve the problem, the following requirements must be met:

- If the objective is minimized, the sum of non-linear terms must be convex, otherwise it must be concave.
- Any constraint bounded below must be concave, and any constraint bounded above must be convex.
- Each separable term must be twice differentiable within the bounds of the variable it is applied to.

Some simple rules can be followed to ensure that the problem satisfies **MOSEK**'s convexity and differentiability requirements. First of all, for any variable $x_i$ used in a separable term, the variable bounds must define a range within which the function is twice differentiable. These bounds are defined in Table 7.2.

Table 7.2: Safe bounds for functions in the SCopt interface.

| Separable function | Operator name | Safe $x$ bounds |
|---|---|---|
| $fx\ln(x)$ | *ent* | $0 < x$. |
| $fe^{gx+h}$ | *exp* | $-\infty < x < \infty$. |
| $f\ln(gx+h)$ | *log* | If $g > 0$: $-h/g < x$. |
| | | If $g < 0$: $x < -h/g$. |
| $f(x+h)^g$ | *pow* | If $g > 0$ and integer: $-\infty < x < \infty$. |
| | | If $g < 0$ and integer: either $-h < x$ or $x < -h$. |
| | | Otherwise: $-h < x$. |

To ensure convexity, we require that each $z_i(x)$ is either a sum of convex terms or a sum of concave terms. Table 7.3 lists convexity conditions for the relevant ranges for $f > 0$ — changing the sign of $f$ switches concavity/convexity.

Table 7.3: Convexity conditions for functions in the SCopt interface.

| Separable function | Operator name | Convexity conditions |
|---|---|---|
| $fx\ln(x)$ | *ent* | Convex within safe bounds. |
| $fe^{gx+h}$ | *exp* | Convex for all $x$. |
| $f\ln(gx+h)$ | *log* | Concave within safe bounds. |
| $f(x+h)^g$ | *pow* | If $g$ is even integer: convex within safe bounds. |
| | | If $g$ is odd integer:<br>• concave if $(-\infty, -h)$,<br>• convex if $(-h, \infty)$<br>If $0 < g < 1$: concave within safe bounds. |
| | | Otherwise: convex within safe bounds. |

A problem involving linear combinations of variables (such as $\ln(x_1+x_2)$), can be converted to a separable problem using slack variables and additional equality constraints.

### 7.1.1 Example

Consider the following separable convex problem:

$$\begin{array}{ll} \text{minimize} & \exp(x_2) - \ln(x_1) \\ \text{subject to} & x_2 \ln(x_2) \leq 0 \\ & x_1^{1/2} - x_2 \geq 0 \\ & \frac{1}{2} \leq x_1, x_2 \leq 1. \end{array} \tag{7.1}$$

Note that all nonlinear functions are well defined for $x$ values satisfying the variable bounds strictly. This assures that function evaluation errors will not occur during the optimization process because **MOSEK**.

The **MOSEK** Toolbox for MATLAB provides a simple interface for separable convex problem called SCopt, and composed by a single function *mskscopt*.

When using the SCopt interface to solve problem (7.1), the linear part of the problem, such as a $c$ and $A$, is specified as usual using MATLAB vectors and matrices. However, the nonlinear functions must be specified using five arrays which in the case of problem (7.1) can have the form:

```
opr  = ['log'; 'exp'; 'ent'; 'pow'];
opri = [0    ;     0;     1;     2];
oprj = [1    ;     2;     2;     1];
oprf = [-1   ;     1;     1;     1];
oprg = [1    ;     1;     0;   0.5];
oprh = [0    ;     0;     0;     0];
```

Hence,

- `opr(k,:)` specifies the type of a nonlinear function,
- `opri(k)` specifies in which constraint the nonlinear function should be added (zero means objective),
- `oprj(k)` means that the nonlinear function should be applied to $x_j$,
- `oprf(k)`, `oprg(k)` and `oprh(k)` are parameters used by the *mskscopt* function according to Table 7.1.

The `i` value indicates which constraint the nonlinear function belongs to. However, if `i` is identical to zero, then the function belongs to the objective.

## 7.2 Entropy Optimization

An entropy optimization problem has the following form

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

where all the components of $d$ must be nonnegative, i.e. $d_j \geq 0$.

### Example

An example of an entropy optimization problem is

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

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

Listing 7.1: Entropy optimization example.

```
function eo1()
d     = [1 1]'
c     = [-1 0]'
a     = [1 1]
blc   = 1
buc   = 1
[res] = mskenopt(d,c,a,blc,buc)
res.sol.itr.xx
```

## 7.3 Geometric Optimization

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

$$\begin{array}{lccl} \text{minimize} & \sum_{k \in J_0} c_k \prod_{j=1}^{n} t_j^{a_{kj}} & & \\ \text{subject to} & \sum_{k \in J_i} c_k \prod_{j=1}^{n} t_j^{a_{kj}} & \leq & 1, \quad i = 1, \ldots, m, \\ & t > 0, & & \end{array} \tag{7.2}$$

where it is assumed that

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

and if $i \neq j$, then

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

Hence, $A$ is a $T \times n$ matrix and $c$ is a vector of length $t$. In general, the problem (7.2) is very hard to solve, but the posynomial case where

$$c > 0$$

is relatively easy. Using the variable transformation

$$t_j = e^{x_j} \tag{7.3}$$

we obtain the problem

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

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

$$\begin{array}{lccl} \text{minimize} & \log(\sum_{k \in J_0} c_k e^{a_{k:} x}) & & \\ \text{subject to} & \log(\sum_{k \in J_i} c_k e^{a_{k:} x}) & \leq & \log(1), \quad i = 1, \ldots, m, \end{array} \tag{7.4}$$

which is also a convex optimization problem since log is strictly increasing. Hence, the problem (7.4) can be solved by **MOSEK**.

For further details about geometric optimization we refer the reader to *[BSS93]*.

**MOSEK** cannot handle a geometric optimization problem directly, but the transformation (7.4) can be solved using the **MOSEK** optimization toolbox function mskgpopt Please note that the solution to the transformed problem can easily be converted into a solution to the original geometric optimization problem using relation (7.3).

Subsequently, we will use the example

$$\begin{array}{lccl} \text{minimize} & 40 t_1^{-1} t_2^{-\frac{1}{2}} t_3^{-1} + 20 t_1 t_3 + 40 t_1 t_2 t_3 & & \\ \text{subject to} & \frac{1}{3} t_1^{-2} t_2^{-2} + \frac{4}{3} t_2^{\frac{1}{2}} t_3^{-1} & \leq & 1, \\ & 0 < t_1, t_2, t_3 & & \end{array} \tag{7.5}$$

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

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

and the vectors

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

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

Listing 7.2 demonstrates how the example is solved using *mskgpopt*.

Listing 7.2: Example on how to use mskgpopt.

```
%%
%
%  Copyright : Copyright (c) MOSEK ApS, Denmark. All rights reserved.
%
%  File :      go1.m
%
%  Purpose : Demonstrates a simple geometric optimization problem.
%
%%

function go1()

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

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

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

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

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

% Dual multipliers (should be negative)
fprintf('Dual variables (should be negative):');
```

```
fprintf(' %e',res.sol.itr.y);
fprintf('\n\n');
```

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

```
max(res.sol.itr.xc) < = 0.0
```

or equivalently

```
exp(max(res.sol.itr.xc)) <= 1.0
```

### Solving large scale problems

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

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

where

$$a_{\texttt{subi}[k],\texttt{subj}[k]} = \texttt{valij}[k].$$

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

```
help sparse
```

in MATLAB.

### Preprocessing tip

Before solving a geometric optimization problem it is worthwhile to check if a column of the $A$ matrix inputted to mskgpopt contains only positive elements. If this is the case, the corresponding variable $t_i$ can take the value zero in the optimal solution: This may cause problems for **MOSEK** so it is better to remove such variables from the problem — doing so will have no influence on the optimal solution.

### Reading and writing problems to a file

The functions *mskgpread* and *mskgpwri* can used to read and write geometric programming problems to file, see the Command Reference in Sec. 17.1.

CHAPTER

EIGHT

# ADVANCED TUTORIALS

## 8.1 Linear Least Squares and Related Norm Minimization Problems

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

$$\text{minimize} \|Fx - b\| \tag{8.1}$$

where $F$ and $b$ are a matrix and vector of appropriate dimensions. $x$ is the vector decision variables. Typically, the norm used is the 1-norm, the 2-norm, or the infinity norm.

### 8.1.1 The Case of the 2-norm

Initially let us focus on the 2 norm. In this case (8.1) is identical to the quadratic optimization problem

$$\text{minimize } \frac{1}{2} x^T F^T F x + \frac{1}{2} b^T b - b^T F x \tag{8.2}$$

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

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

Subsequently, it is demonstrated how the quadratic optimization problem (8.2) is solved using *mosekopt*. In the example the problem data is read from a file, then data for the problem (8.2) is constructed and finally the problem is solved.

Listing 8.1: Script solving problem (8.2)

```
function nrm1()
% Clear prob
clear prob;

F = [ [ 0.4302 , 0.3516 ]; [0.6246, 0.3384] ]
b = [ 0.6593, 0.9666]'

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

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

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

% Obtain the matrix dimensions.
```

```
[m,n]   = size(F);

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

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

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

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

Often the $x$ variables must be within some bounds or satisfy some additional linear constraints. These requirements can easily be incorporated into the problem (8.2). E.g. the constraint $\|x\|_\infty \leq 1$ can be modeled as reported in Listing 8.2.

Listing 8.2: Script solving an extension of problem (8.2)

```
function nrm2()

F = [ [ 0.4302 , 0.3516 ]; [0.6246, 0.3384] ]
b = [ 0.6593, 0.9666]'

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

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

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

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

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

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

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

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

### 8.1.2 The Case of the Infinity Norm

In some applications of the norm minimization problem (8.1) it is better to use the infinity norm than the 2 norm. However, the problem (8.1) stated as an infinity norm problem is equivalent to the linear optimization problem

$$\begin{array}{lrcl} \text{minimize} & \tau & & \\ \text{subject to} & Fx + \tau e - b & \geq & 0, \\ & Fx - \tau e - b & \leq & 0, \end{array} \tag{8.3}$$

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

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

and hence at optimum

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

holds. Problem (8.3) is straightforward to solve, for instance using script as in Listing 8.3

Listing 8.3: Script solving problem (8.3).

```
function nrm3()
clear prob;


F = [ [ 0.4302 , 0.3516 ]; [0.6246, 0.3384] ]
b = [ 0.6593, 0.9666]'


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

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

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

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

### 8.1.3 The Case of the 1-norm

By definition, for the 1-norm we have that

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

Therefore, the norm minimization problem can be formulated as follows

$$\begin{array}{lrcl} \text{minimize} & \sum_{i=1}^m t_i & & \\ \text{subject to} & |f_{i:}x - b_i| & = & t_i, \quad i = 1,\ldots,m, \end{array}$$

which in turn is equivalent to

$$\begin{array}{lrcl} \text{minimize} & \sum_{i=1}^m t_i & & \\ \text{subject to} & f_{i:}x - b_i & \leq & t_i, \quad i = 1,\ldots,m, \\ & -(f_{i:}x - b_i) & \leq & t_i, \quad i = 1,\ldots,m. \end{array}$$

The reader should verify that this is really the case. In matrix notation this problem can be expressed as follows

$$\begin{array}{lrcl} \text{minimize} & e^T t & & \\ \text{subject to} & Fx - te & \leq & b, \\ & Fx + te & \geq & b, \end{array} \tag{8.4}$$

where $e = (1, \ldots, 1)^T$. Next, this problem is solved in Listing 8.4.

Listing 8.4: Script solving problem (8.4).

```
function nrm4()
clear prob;
```

```
F = [ [ 0.4302 , 0.3516 ]; [0.6246, 0.3384] ]
b = [ 0.6593, 0.9666]'

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

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

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

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

#### A better formulation

It is possible to improve upon the formulation of the problem (8.3). Indeed problem (8.3) is equivalent to

$$\begin{array}{lccll} \text{minimize} & \sum_{i=1}^{m} t_i & & & \\ \text{subject to} & f_{i:}x - b_i - t_i + v_i & = & 0, & i = 1,\ldots,m, \\ & -(f_{i:}x - b_i) - t_i & \leq & 0, & i = 1,\ldots,m, \\ & v_i \geq 0, & & & i = 1,\ldots,m. \end{array} \tag{8.5}$$

After eliminating the $t$ variables then this problem is equivalent to

$$\begin{array}{lccll} \text{minimize} & \sum_{i=1}^{m} (f_{i:}x - b_i + v_i) & & & \\ \text{subject to} & -2(f_{i:}x - b_i) - v_i & \leq & 0, & i = 1,\ldots,m, \\ & v_i \geq 0, & & & i = 1,\ldots,m. \end{array} \tag{8.6}$$

Please note that this problem has only half the number of general constraints than problem (8.3) since we have replaced constraints of the general form

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

with simpler constraints

$$v_i \geq 0$$

which **MOSEK** treats in a special and highly efficient way. Furthermore **MOSEK** stores only the non-zeros in the coefficient matrix of the constraints. This implies that the problem (8.6) is likely to require much less space than the problem (8.5).

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

#### More About Solving Linear Least Squares Problems

Linear least squares problems with and without linear side constraints appear very frequently in practice and it is therefore important to know how such problems are solved efficiently using **MOSEK**. Now, assume that the problem of interest is the linear least squares problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|Fx - f\|_2^2 \\ \text{subject to} & Ax = b, \\ & l^x \leq x \leq u^x, \end{array} \tag{8.7}$$

where $F$ and $A$ are matrices and the remaining quantities are vectors. $x$ is the vector of decision variables. The problem (8.7) as stated is a convex quadratic optimization problem and can be solved as such.

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

$$\begin{array}{lccc}\text{minimize} & \frac{1}{2}\|z\|_2^2 & & \\ \text{subject to} & Ax & = & b, \\ & Fx - z & = & f, \\ & l^x \leq x \leq u^x. & & \end{array} \tag{8.8}$$

Please note that a number of new constraints and variables has been introduced which of course seems to be disadvantageous but on the other hand the Hessian of the objective in problem (8.8) is much sparser than in problem (8.7). Frequently this turns out to be more important for the computational efficiency and therefore the latter formulation is usually the better one.

If $F$ has many more rows than columns, then formulation (8.8) is not attractive whereas the corresponding dual problem is. Using the duality theory outlined in Sec. 16.5.1 we obtain the dual problem

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

which can be simplified to

$$\begin{array}{lc}\text{maximize} & b^T y + f^T z + (l^x)^T s_l^x + (u^x)^T s_u^x - \frac{1}{2}\|z\|_2^2 \\ \text{subject to} & A^T y + F^T z + s_l^x - s_u^x = 0, \\ & s_l^x, s_u^x \geq 0 \end{array} \tag{8.9}$$

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

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

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

Given our assumptions the dual problem (8.9) will have much fewer constraints than the primal problem (8.8); in general, the fewer constraints a problem contains, the more efficient **MOSEK** tends to be. A question is: If the dual problem (8.9) is solved instead of the primal problem (8.8), how is the optimal $x$ solution obtained? It turns out that the dual variables corresponding to the constraint

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

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

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

vector, the optimal $x$ solution can easily be obtained.

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

Listing 8.5: Comparison on whether the primal or the dual is more attractive to solve.

```
function nrm5()
F = repmat( [ [ 0.4302, 0.3516 ]; [0.6246, 0.3384] ], 10, 1);
f = repmat( [ 0.6593, 0.9666]', 10,1) ;
% Obtain the matrix dimensions.
[m,n]   = size(F)

prob        = [];

prob.qosubi = n+(1:m);
prob.qosubj = n+(1:m);
```

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

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

fprintf('First try\n');

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

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

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

  % objective value:
  fprintf('Objective value: %-6e\n', 0.5*norm(F*x(1:n)-f)^2);

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

% Clear prob.
prob=[];

%
% Next, we solve the dual problem.

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

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

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

fprintf('\n\nSecond try\n');
tic
```

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

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

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

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

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

Here is the output produced:

Listing 8.6: Output of *nrm5.m*.

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

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

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

### 8.1.4 Using Conic Optimization on Linear Least Squares Problems

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

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

is equivalent to

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

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

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

which can be reduced to

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

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

## 8.2 Converting a quadratically constrained problem to conic form

**MOSEK** employs the following form of quadratic problems:

$$
\begin{array}{lccccll}
\text{minimize} & & & \frac{1}{2} x^T Q^o x + c^T x + c^f & & & \\
\text{subject to} & l_k^c & \leq & \frac{1}{2} x^T Q^k x + \sum_{j=0}^{n-1} a_{k,j} x_j & \leq & u_k^c, & k = 0, \ldots, m-1, \\
 & l_j^x & \leq & x_j & \leq & u_j^x, & j = 0, \ldots, n-1.
\end{array}
\tag{8.10}
$$

A conic quadratic constraint has the form

$$
x \in \mathcal{Q}^n
$$

in its most basic form where

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

A quadratic problem such as (8.10), if convex, can be reformulated in conic form. This is in fact the reformulation **MOSEK** performs internally. It has many advantages:

- elegant duality theory for conic problems,
- reporting accurate dual information for quadratic inequalities is hard and/or computational expensive,
- it certifies that the original quadratic problem is indeed convex,
- modelling directly in conic form usually leads to a better model *[And13]* i.e. a faster solution time and better numerical properties.

In addition, there are more types of conic constraints that can be combined with a quadratic cone, for example semidefinite cones.

**MOSEK** offers a function that performs the conversion from quadratic to conic quadratic form explicitly. Note that the reformulation is not unique. The approach followed by **MOSEK** is to introduce additional variables, linear constraints and quadratic cones to obtain a larger but equivalent problem in which the original variables are preserved.

In particular:

- all variables and constraints are kept in the problem,
- each quadratic constraint and quadratic terms in the objective generate one rotated quadratic cone,
- each quadratic constraint will contain no coefficients and upper/lower bounds will be set to $\infty, -\infty$ respectively.

This allows the user to recover the original variable and constraint values, as well as their dual values, with no conversion or additional effort.

### 8.2.1 Quadratic Constraint Reformulation

Let us assume we want to convert the following quadratic constraint

$$l \leq \frac{1}{2}x^T Qx + \sum_{j=0}^{n-1} a_j x_j \leq u$$

to conic form. We first check whether $l = -\infty$ or $u = \infty$, otherwise either the constraint can be dropped, or the constraint is not convex. Thus let us consider the case

$$\frac{1}{2}x^T Qx + \sum_{j=0}^{n-1} a_j^T x_j \leq u. \tag{8.11}$$

Introducing an additional variable $w$ such that

$$w = u - \sum_{j=0}^{n-1} a_j^T x_j \tag{8.12}$$

we obtain the equivalent form

$$\begin{array}{lcl} \frac{1}{2}x^T Qx & \leq & w, \\ u - \sum_{j=0}^{n-1} a_j x_j & = & w. \end{array}$$

If $Q$ is positive semidefinite, then there exists a matrix $F$ such that

$$Q = FF^T \tag{8.13}$$

and therefore we can write

$$\begin{array}{lcl} \|Fx\|^2 & \leq & 2w, \\ u - \sum_{j=0}^{n-1} a_j^T x_j & = & w. \end{array}$$

Introducing an additional variable $z = 1$, and setting $y = Fx$ we obtain the conic formulation

$$\begin{array}{ll} (w, z, y) & \in \mathcal{Q}_r, \\ z = 1 & \\ y = Fx & \\ w = u - a^T x. & \end{array} \tag{8.14}$$

Summarizing, for each quadratic constraint involving $t$ variables, **MOSEK** introduces

1. a rotated quadratic cone of dimension $t + 2$,
2. two additional variables for the cone roots,
3. $t$ additional variables to map the remaining part of the cone,
4. $t$ linear constraints.

A quadratic term in the objective is reformulated in a similar fashion. We refer to *[And13]* for a more thorough discussion.

### Example

Next we consider a simple problem with quadratic objective function:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}(13x_0^2 + 17x_1^2 + 12x_2^2 + 24x_0x_1 + 12x_1x_2 - 4x_0x_2) - 22x_0 - 14.5x_1 + 12x_2 + 1 \\ \text{subject to} & -1 \le x_0, x_1, x_2 \le 1 \end{array}$$

We can specify it in the human-readable OPF format.

```
[comment]
An example of small QO problem from Boyd and Vandenberghe, "Convex Optimization", page 189 ex␣
↪4.3
The solution is (1,0.5,-1)
[/comment]

[variables]
x0 x1 x2
[/variables]

[objective min]
0.5 (13 x0^2  + 17 x1^2 + 12 x2^2 + 24 x0 * x1 + 12 x1 * x2 - 4 x0 * x2 ) - 22 x0 - 14.5 x1 +␣
↪12 x2 + 1
[/objective]

[bounds]
[b] -1 <= * <= 1 [/b]
[/bounds]
```

The objective function is convex, the minimum is attained for $x^\star = (1, 0.5, -1)$. The conversion will introduce first a variable $x_3$ in the objective function such that $x_3 \ge 1/2x^TQx$ and then convert the latter directly in conic form. The converted problem follows:

$$\begin{array}{lr} \text{minimize} & -22x_0 - 14.5x_1 + 12x_2 + x_3 + 1 \\ \text{subject to} & 3.61x_0 + 3.33x_1 - 0.55x_2 - x_6 = 0 \\ & +2.29x_1 + 3.42x_2 - x_7 = 0 \\ & 0.81x_1 - x_8 = 0 \\ & -x_3 + x_4 = 0 \\ & x_5 = 1 \\ & (x_4, x_5, x_6, x_7, x_8) \in \mathcal{Q}_\nabla \\ & -1 \le x_0, x_1, x_2 \le 1 \end{array}$$

We obtain the reformulation as follows:

```
[r, res] = mosekopt('read(quad.opf)')
probQuad = res.prob
[r, res2] = mosekopt('toconic prob', probQuad)
probConic = res2.prob
mosekopt('write(conic.opf)', probConic)
```

and the output is:

```
[comment]
   Written by MOSEK version 8.1.0.19
   Date 21-08-17
   Time 10:53:36
[/comment]

[hints]
  [hint NUMVAR] 9 [/hint]
  [hint NUMCON] 4 [/hint]
  [hint NUMANZ] 11 [/hint]
  [hint NUMQNZ] 0 [/hint]
```

```
  [hint NUMCONE] 1 [/hint]
[/hints]

[variables disallow_new_variables]
  x0000_x0 x0001_x1 x0002_x2 x0003 x0004
  x0005 x0006 x0007 x0008
[/variables]

[objective minimize]
   - 2.2e+01 x0000_x0 - 1.45e+01 x0001_x1 + 1.2e+01 x0002_x2 + x0003
   + 1e+00
[/objective]

[constraints]
  [con c0000]  3.605551275463989e+00 x0000_x0 - 5.547001962252291e-01 x0002_x2 + 3.
↪328201177351375e+00 x0001_x1 - x0006 = 0e+00 [/con]
  [con c0001]  3.419401657060442e+00 x0002_x2 + 2.294598480395823e+00 x0001_x1 - x0007 = 0e+00␣
↪[/con]
  [con c0002]  8.111071056538127e-01 x0001_x1 - x0008 = 0e+00 [/con]
  [con c0003]  - x0003 + x0004 = 0e+00 [/con]
[/constraints]

[bounds]
  [b] -1e+00     <= x0000_x0,x0001_x1,x0002_x2 <= 1e+00 [/b]
  [b]               x0003,x0004 free [/b]
  [b]               x0005 =  1e+00 [/b]
  [b]               x0006,x0007,x0008 free [/b]
  [cone rquad k0000] x0004, x0005, x0006, x0007, x0008 [/cone]
[/bounds]
```

We can clearly see that constraints c0000, c0001 and c0002 represent the original linear constraints as in (8.13), while c0003 corresponds to (8.12). The cone roots are x0005 and x0004.

CHAPTER

NINE

# CASE STUDIES

In this section we present some case studies in which the Optimization Toolbox for MATLAB is used to solve real-life applications. These examples involve some more advanced modelling skills and possibly some input data. The user is strongly recommended to first read the basic tutorials of Sec. 6 before going through these advanced case studies.

| Case Studies | Type | Int. | Keywords |
|---|---|---|---|
| *Robust linear optimization* | CQO | NO | Robust optimization |
| *Geometric optimization* | EXPOPT | NO | Polynomial optimization |

## 9.1 Robust linear Optimization

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

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

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

This section is co-authored with A. Ben-Tal and A. Nemirovski. For further information about robust linear optimization consult *[BTN00]*, *[BenTalN01]*.

### 9.1.1 Introductory Example

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

| | | |
|---|---|---|
| Selling price \$ per 1000 packs | 6200 | 6900 |
| Content of agent A gm per 100 packs | 0.500 | 0.600 |
| Production expenses | | |
| \$ per 1000 packs | | |
| Manpower, hours | 90.0 | 100.0 |
| Equipment, hours | 40.0 | 50.0 |
| Operational cost, \$ | 700 | 800 |

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

| Raw material | Purchasing price, | Content of agent A, |
|---|---|---|
| RawI | 100.00 | 0.01 |
| RawII | 199.90 | 0.02 |

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

| Budget,' | Manpower | Equipment | Capacity of raw materials |
|---|---|---|---|
| 100000 | 2000 | 800 | 1000 |

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

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

and maximize the income

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

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

Minimize

$$-\{100 \cdot \mathtt{RawI} + 199.90 \cdot \mathtt{RawII} + 700 \cdot \mathtt{DrugI} + 800 \cdot \mathtt{DrugII}\} + \{6200 \cdot \mathtt{DrugI} + 6900 \cdot \mathtt{DrugII}\} \tag{9.1}$$

subject to

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

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

- Balance of the active agent.
- Storage restriction.
- Manpower restriction.
- Equipment restriction.
- Ducget restriction.

Listing 9.1 is the MATLAB script which specifies the problem and solves it using the **MOSEK** optimization toolbox:

Listing 9.1: Script *rlo1.m*.

```
function rlo1()

prob.c   = [-100;-199.9;6200-700;6900-800];
prob.a   = sparse([0.01,0.02,-0.500,-0.600;1,1,0,0;
                   0,0,90.0,100.0;0,0,40.0,50.0;100.0,199.9,700,800]);
prob.blc = [0;-inf;-inf;-inf;-inf];
prob.buc = [inf;1000;2000;800;100000];
prob.blx = [0;0;0;0];
prob.bux = [inf;inf;inf;inf];
[r,res]  = mosekopt('maximize',prob);
```

```
xx      = res.sol.itr.xx;
RawI    = xx(1);
RawII   = xx(2);
DrugI   = xx(3);
DrugII  = xx(4);

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

When executing this script, the following is displayed:

Listing 9.2: Output of script *rlo1.m*

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

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

## 9.1.2 Data Uncertainty and its Consequences.

Please note that not all problem data can be regarded as *absolutely* reliable; e.g. one can hardly believe that the contents of the active agent in the raw materials are *exactly* the *nominal data* 0.01 gm/kg for `RawI` and 0.02 gm/kg for *RawII*. In reality, these contents definitely vary around the indicated values. A natural assumption here is that the actual contents of the active agent $a_i$ in `RawI` and $a_{II}$ in `RawII` are realizations of random variables somehow distributed around the *nominal contents* $a_i^{\mathrm{n}} = 0.01$ and $a_{II}^{\mathrm{n}} = 0.02$. To be more specific, assume that $a_i$ drifts in the 0.5% margin of $a_i^{\mathrm{n}}$, i.e. it takes with probability 0.5 the values from the interval $a_i^{\mathrm{n}}(1 \pm 0.005) = a_i^{\mathrm{n}}\{0.00995; 0.01005\}$. Similarly, assume that $a_{II}$ drifts in the 2% margin of $a_{II}^{\mathrm{n}}$, taking with probabilities 0.5 the values $a_{II}^{\mathrm{n}}(1 \pm 0.02) = a_i^{\mathrm{n}}\{0.0196; 0.0204\}$. How do the perturbations of the contents of the active agent affect the production process?

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

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

It turns out that the outlined phenomenon is found in many linear programs of practical origin. Usually, in these programs at least part of the data is not known exactly and can vary around its nominal values, and these data perturbations can make the nominal optimal solution – the one corresponding to the nominal data – infeasible. It turns out that the consequences of data uncertainty can be much more severe than in our toy example. The analysis of linear optimization problems from the NETLIB collection[1] reported in *[BTN00]* demonstrates that for 13 of 94 NETLIB problems, already 0.01% perturbations of "clearly uncertain" data can make the nominal optimal solution severely infeasible: with these perturbations, the solution, with a non-negligible probability, violates some of the constraints by 50% and more. It should be added that in the general case, in contrast to the toy example we have considered, there is no evident way to adjust the optimal solution by a small modification to the actual values of the data. Moreover there are cases when such an adjustment is impossible — in order to become feasible for the perturbed data, the nominal optimal solution should be *completely reshaped*.

## 9.1.3 Robust Linear Optimization Methodology

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

### Uncertain Linear Programs and their Robust Counterparts.

Consider a linear optimization problem

$$
\begin{array}{lccccc}
\text{minimize} & & & c^T x & & \\
\text{subject to} & l_c & \leq & Ax & \leq & u_c, \\
 & l_x & \leq & x & \leq & u_x,
\end{array}
\tag{9.2}
$$

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

$$
\begin{array}{rcl}
\mathcal{U} & = & \{(c, A, l_c, u_c, l_x, u_x) : \\
 & & (c^{\mathrm{n}} - dc, A^{\mathrm{n}} - dA, l_c^{\mathrm{n}} - dl_c, u_c^{\mathrm{n}} - du_c, l_x^{\mathrm{n}} - dl_x, u_x^{\mathrm{n}} - du_x) \leq (c, A, l_c, u_c, l_x, u_x) \\
 & & \leq (c^{\mathrm{n}} + dc, A^{\mathrm{n}} + dA, l_c^{\mathrm{n}} + dl_c, u_c^{\mathrm{n}} + du_c, l_x^{\mathrm{n}} + dl_x, u_x^{\mathrm{n}} + du_x)\}.
\end{array}
\tag{9.3}
$$

Here

$$(c^{\mathrm{n}}, A^{\mathrm{n}}, l_c^{\mathrm{n}}, u_c^{\mathrm{n}}, l_x^{\mathrm{n}}, u_x^{\mathrm{n}})$$

is the *nominal data*,

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

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

- The family of instances (9.2) with data running through a given uncertainty set $\mathcal{U}$ is called an *uncertain linear optimization problem*.
- Vector $x$ is called a *robust feasible solution* to an uncertain linear optimization problem, if it remains feasible for all realizations of the data from the uncertainty set, i.e. if

$$l_c \leq Ax \leq u_c, l_x \leq x \leq u_x$$

for all

$$(c, A, l_c, u_c, l_x, u_x) \in \mathcal{U}.$$

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

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

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

$$\min_{t,x} \left\{ t : c^T x \leq t, l_c \leq Ax \leq u_c, l_x \leq x \leq u_x \forall (c, A, l_c, u_c, l_x, u_x) \in \mathcal{U} \right\}. \tag{9.4}$$

The optimal solution to (9.4) is treated as the *uncertainty-immuned* solution to the original uncertain linear programming program.

### Robust Counterpart of an Uncertain Linear Optimization Problem with Interval Uncertainty

In general, the RC (9.4) of an uncertain linear optimization problem is not a linear optimization problem since (9.4) has infinitely many linear constraints. There are, however, cases when (9.4) can be rewritten equivalently as a linear programming program; in particular, this is the case for interval uncertainty (9.3). Specifically, in the case of (9.3), the robust counterpart of uncertain linear program is equivalent to the following linear program in variables $x, y, t$:

$$\begin{array}{lccccll}
\text{minimize} & & & t & & & \\
\text{subject to} & & & (c^{\mathrm{n}})^T x + (dc)^T y - t & \leq & 0, & (a) \\
& l_c^{\mathrm{n}} + dl_c & \leq & (A^{\mathrm{n}})x - (dA)y, & & & (b) \\
& & & (A^{\mathrm{n}})x + (dA)y & \leq & u_c^{\mathrm{n}} - du_c, & (c) \\
& 0 & \leq & x + y, & & & (d) \\
& 0 & \leq & -x + y, & & & (e) \\
& l_x^{\mathrm{n}} + dl_x & \leq & x & \leq & u_x^{\mathrm{n}} - du_x, & (f)
\end{array} \tag{9.5}$$

The origin of (9.5) is quite transparent: The constraints $d - e$ in (9.5) linking $x$ and $y$ merely say that $y_i \geq |x_i|$ for all $i$. With this in mind, it is evident that at every feasible solution to (9.5) the entries in the vector

$$(A^{\mathrm{n}})x - (dA)y$$

are lower bounds on the entries of $Ax$ with $A$ from the uncertainty set (9.3), so that $(b)$ in (9.5) ensures that $l_c \leq Ax$ for all data from the uncertainty set. Similarly, $(c), (a)$ ans $f$ in (9.5) ensure, for all data from the uncertainty set, that $Ax \leq u_c$, $c^T x \leq t$, and that the entries in $x$ satisfy the required lower and upper bounds, respectively.

Please note that at the optimal solution to (9.5), one clearly has $y_j = |x_j|$. It follows that when the bounds on the entries of $x$ impose nonnegativity (nonpositivity) of an entry $x_j$, then there is no need to introduce the corresponding additional variable $y_i$ — from the very beginning it can be replaced with $x_j$, if $x_j$ is nonnegative, or with $-x_j$, if $x_j$ is nonpositive.

Another possible formulation of problem (9.5) is the following. Let

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

then this equation is equivalent to $(a) - (b)$ in (9.5). If $(l_c)_i = -\infty$, then equation $i$ should be dropped from the computations. Similarly,

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

is equivalent to $(d)$ in (9.5). This implies that

$$l_c^{\mathrm{n}} + dl_c - (A^{\mathrm{n}})x + f = -(dA)y$$

and that

$$y = g + x$$

Substituting these values into (9.5) gives

$$\begin{array}{lllll}
\text{minimize} & & t & & \\
\text{subject to} & & (c^{\mathrm{n}})^T x + (dc)^T (g+x) - t & \leq & 0, \\
0 & \leq & f, & & \\
 & & 2(A^{\mathrm{n}})x + (dA)(g+x) + f + l_c^{\mathrm{n}} + dl_c & \leq & u_c^{\mathrm{n}} - du_c, \\
0 & \leq & g, & & \\
0 & \leq & 2x + g, & & \\
l_x^{\mathrm{n}} + dl_x & \leq & x & \leq & u_x^{\mathrm{n}} - du_x,
\end{array}$$

which after some simplifications leads to

$$\begin{array}{llllll}
\text{minimize} & & t & & & \\
\text{subject to} & & (c^{\mathrm{n}} + dc)^T x + (dc)^T g - t & \leq & 0, & (a) \\
0 & \leq & f, & & & (b) \\
 & & 2(A^{\mathrm{n}} + dA)x + (dA)g + f - (l_c^{\mathrm{n}} + dl_c) & \leq & u_c^{\mathrm{n}} - du_c, & (c) \\
0 & \leq & g, & & & (d) \\
0 & \leq & 2x + g, & & & (e) \\
l_x^{\mathrm{n}} + dl_x & \leq & x & \leq & u_x^{\mathrm{n}} - du_x, & (f)
\end{array}$$

and

$$\begin{array}{llllll}
\text{minimize} & & t & & & \\
\text{subject to} & & (c^{\mathrm{n}} + dc)^T x + (dc)^T g - t & \leq & 0, & (a) \\
 & & 2(A^{\mathrm{n}} + dA)x + (dA)g + f & \leq & u_c^{\mathrm{n}} - du_c + l_c^{\mathrm{n}} + dl_c, & (b) \\
0 & \leq & 2x + g, & & & (c) \\
0 & \leq & f, & & & (d) \\
0 & \leq & g, & & & (e) \\
l_x^{\mathrm{n}} + dl_x & \leq & x & \leq & u_x^{\mathrm{n}} - du_x. & (f)
\end{array} \tag{9.6}$$

Please note that this problem has more variables but much fewer constraints than (9.5). Therefore, (9.6) is likely to be solved faster than (9.5). Note too that (9.6).$b$ is trivially redundant if $l_x^{\mathrm{n}} + dl_x \geq 0$.

### Introductory Example (continued)

Let us apply the Robust Optimization methodology to our drug production example presented in Sec. 9.1.1, assuming that the only uncertain data is the contents of the active agent in the raw materials, and that these contents vary in 0.5% and 2% neighborhoods of the respective nominal values 0.01 and 0.02. With this assumption, the problem becomes an uncertain LP affected by interval uncertainty; the robust counterpart (9.5) of this uncertain LP is the linear program

$$\begin{array}{lll}
(\text{Drug_RC}): & & \\
\text{maximize} & & \\
t & & \\
\text{subject to} & & \\
t \leq -100 \cdot \mathtt{RawI} - 199.9 \cdot \mathtt{RawII} + 5500 \cdot \mathtt{DrugI} + 6100 \cdot \mathtt{DrugII} & & \\
0.01 \cdot 0.995 \cdot \mathtt{RawI} + 0.02 \cdot 0.98 \cdot \mathtt{RawII} - 0.500 \cdot \mathtt{DrugI} - 0.600 \cdot \mathtt{DrugII} & \geq & 0 \\
\mathtt{RawI} + \mathtt{RawII} & \leq & 1000 \\
90.0 \cdot \mathtt{DrugI} + 100.0 \cdot \mathtt{DrugII} & \leq & 2000 \\
40.0 \cdot \mathtt{DrugI} + 50.0 \cdot \mathtt{DrugII} & \leq & 800 \\
100.0 \cdot \mathtt{RawI} + 199.90 \cdot \mathtt{RawII} + 700 \cdot \mathtt{DrugI} + 800 \cdot \mathtt{DrugII} & \leq & 100000 \\
\mathtt{RawI}, \mathtt{RawII}, \mathtt{DrugI}, \mathtt{DrugII} & \geq & 0
\end{array} \tag{9.7}$$

Solving this problem with **MOSEK** we get the following output:

Listing 9.3: Output solving problem (9.7).

```
*** Optimal value: 8294.567
*** Optimal solution:
RawI:    877.732
```

```
RawII:     0.000
DrugI:    17.467
DrugII:    0.000
```

We see that the robust optimal solution we have built *costs money* – it promises a profit of just $8,295$ (cf. with the profit of $8,820$ promised by the nominal optimal solution). Please note, however, that the robust optimal solution remains feasible whatever are the realizations of the uncertain data from the uncertainty set in question, while the nominal optimal solution requires adjustment to this data and, with this adjustment, results in the average profit of $7,843$, which is by $5.4\%$ *less* than the profit of ‘ 8,295 *guaranteed* by the robust optimal solution. Note too that the robust optimal solution is significantly different from the nominal one: both solutions prescribe to produce the same drug `DrugI` (in the amounts $17,467$ and $17,552$ packs, respectively) but from different raw materials, `RawI` in the case of the robust solution and `RawII` in the case of the nominal solution. The reason is that although the price per unit of the active agent for `RawII` is sligthly less than for `RawI`, the content of the agent in `RawI` is more stable, so when possible fluctuations of the contents are taken into account, `RawI` turns out to be more profitable than `RawII`.

### 9.1.4 Random Uncertainty and Ellipsoidal Robust Counterpart

In some cases, it is natural to assume that the perturbations affecting different uncertain data entries are random and independent of each other. In these cases, the robust counterpart based on the interval model of uncertainty seems to be too conservative: Why should we expect that all the data will be simultaneously driven to its most unfavorable values and immune the solution against this highly unlikely situation? A less conservative approach is offered by the *ellipsoidal* model of uncertainty. To motivate this model, let us seseee what happens with a particular linear constraint

$$a^T x \leq b \tag{9.8}$$

at a given candidate solution $x$ in the case when the vector $a$ of coefficients of the constraint is affected by random perturbations:

$$a = a^{\mathrm{n}} + \zeta, \tag{9.9}$$

where $a^{\mathrm{n}}$ is the vector of nominal coefficients and $\zeta$ is a random perturbation vector with zero mean and covariance matrix $V_a$. In this case the value of the left-hand side of (9.8), evaluated at a given $x$, becomes a random variable with the expected value $(a^{\mathrm{n}})^T x$ and the standard deviation $\sqrt{x^T V_a x}$. Now let us act as an engineer who believes that the value of a random variable never exceeds its mean plus 3 times the standard deviation; we do not intend to be that specific and replace *3* in the above rule by a safety parameter $\Omega$ which will be in our control. Believing that the value of a random variable *never* exceeds its mean plus $\Omega$ times the standard deviation, we conclude that a *safe* version of (9.8) is the inequality

$$(a^{\mathrm{n}})^T x + \Omega\sqrt{x^T V_a x} \leq b. \tag{9.10}$$

The word *safe* above admits a quantitative interpretation: If $x$ satisfies (9.10), one can bound from above the probability of the event that random perturbations (9.9) result in violating the constraint (9.8) evaluated at $x$. The bound in question depends on what we know about the distribution of $\zeta$, e.g.

- We always have the bound given by the Tschebyshev inequality: $x$ satisfies (9.10) $\Rightarrow$

$$\mathrm{Prob}\left\{a^T x > b\right\} \leq \frac{1}{\Omega^2}.$$

- When $\zeta$ is Gaussian, then the Tschebyshev bound can be improved to: $x$ satisfies (9.10) $\Rightarrow$

$$\mathrm{Prob}\left\{a^T x > b\right\} \leq \frac{1}{\sqrt{2\pi}}\int_{\Omega}^{\infty} \exp\{-t^2/2\}dt \leq 0.5\exp\{-\Omega^2/2\}. \tag{9.11}$$

- Assume that $\zeta = D\xi$, where $\Delta$ is certain $n \times m$ matrix, and $\xi = (\xi_1, \ldots, \xi_m)^T$ is a random vector with independent coordinates $\xi_1, \ldots, \xi_m$ symmetrically distributed in the segment $[-1, 1]$. Setting $V = DD^T$ ($V$ is a natural *upper bound* on the covariance matrix of $\zeta$), one has: $x$ satisfies (9.10) implies

$$\text{Prob}\left\{a^T x > b\right\} \leq 0.5 \exp\{-\Omega^2/2\}. \tag{9.12}$$

Please note that in order to ensure the bounds in (9.11) and (9.12)) to be $\leq 10^{-6}$, it suffices to set $\Omega = 5.13$.

Now, assume that we are given a linear program affected by random perturbations:

$$\begin{array}{lccccl} \text{minimize} & & & [c^{\text{n}} + dc]^T x & & \\ \text{subject to} & (l_c)_i & \leq & [a_i^{\text{n}} + da_i]^T x & \leq & (u_c)_i, i = 1, \ldots, m, \\ & l_x & \leq & x & \leq & u_x, \end{array} \tag{9.13}$$

where $(c^{\text{n}}, \{a_i^{\text{n}}\}_{i=1}^m, l_c, u_c, l_x, u_x)$ are the nominal data, and $dc$, $da_i$ are random perturbations with zero means[3]. Assume, for the sake of definiteness, that every one of the random perturbations $dc, da_1, \ldots, da_m$ satisfies either the assumption of item 2 or the assumption of item 3, and let $V_c, V_1, \ldots, V_m$ be the corresponding (upper bounds on the) covariance matrices of the perturbations. Choosing a safety parameter $\Omega$ and replacing the objective and the bodies of all the constraints by their safe bounds as explained above, we arrive at the following optimization problem:

$$\begin{array}{lccccl} \text{minimize} & & & t & & \\ \text{subject to} & & & [c^{\text{n}}]^T x + \Omega\sqrt{x^T V_c x} & \leq & t, \\ & (l_c)_i & \leq & [a_i^{\text{n}}]^T x - \Omega\sqrt{x^T V_{a_i} x}, & & \\ & & & [a_i^{\text{n}}]^T x + \Omega\sqrt{x^T V_{a_i} x} & \leq & (u_c)_i, i = 1, \ldots, m, \\ & l_x & \leq & x & \leq & u_x. \end{array} \tag{9.14}$$

The relation between problems (9.14) and (9.13) is as follows:

- If $(x, t)$ is a feasible solution of (9.14), then with probability at least

$$p = 1 - (m + 1) \exp\{-\Omega^2/2\}$$

$x$ is feasible for randomly perturbed problem (9.13), and $t$ is an upper bound on the objective of (9.13) evaluated at $x$.

- We see that if $\Omega$ is not too small (9.14) can be treated as a "safe version" of (9.13).

On the other hand, it is easily seen that (9.14) is nothing but the robust counterpart of the uncertain linear optimization problem with the nominal data $(c^{\text{n}}, \{a_i^{\text{n}}\}_{i=1}^m, l_c, u_c, l_x, u_x)$ and the row-wise ellipsoidal uncertainty given by the matrices $V_c, V_{a_1}, \ldots, V_{a_m}$. In the corresponding uncertainty set, the uncertainty affects the coefficients of the objective and the constraint matrix only, and the perturbation vectors affecting the objective and the vectors of coefficients of the linear constraints run, independently of each other, through the respective ellipsoids

$$\begin{array}{rl} E_c & = \\ \left\{dc = \Omega V_c^{1/2} u : u^T u \leq 1\right\} E_{a_i} & = \\ \left\{da_i = \Omega V_{a_i}^{1/2} u : u^T u \leq 1\right\}, i = 1, \ldots, m. & \end{array}$$

It turns out that in many cases the ellipsoidal model of uncertainty is significantly less conservative and thus better suited for practice, than the interval model of uncertainty.

[3] For the sake of simplicity, we assume that the bounds $l_c, u_c, l_x, u_x$ are not affected by uncertainty; extensions to the case when it is not so are evident.

Last but not least, it should be mentioned that problem (9.14) is equivalent to a conic quadratic program, specifically to the program

$$
\begin{array}{lrcll}
\text{minimize} & & & t & \\
\text{subject to} & & & [c^{\mathrm{n}}]^T x + \Omega z & \leq t, \\
& (l_c)_i & \leq & [a_i^{\mathrm{n}}]^T x - \Omega z_i, & \\
& & & [a_i^{\mathrm{n}}]^T x + \Omega z_i & \leq (u_c)_i, i = 1, \ldots, m, \\
& 0 & = & w - D_c x & \\
& 0 & = & w^i - D_{a_i} x, & i = 1, \ldots, m, \\
& 0 & \leq & z - \sqrt{w^T w}, & \\
& 0 & \leq & z_i - \sqrt{(w^i)^T w^i}, & i = 1, \ldots, m, \\
& l_x & \leq & x & \leq u_x.
\end{array}
$$

where $D_c$ and $D_{a_i}$ are matrices satisfying the relations

$$V_c = D_c^T D_c, V_{a_i} = D_{a_i}^T D_{a_i}, i = 1, \ldots, m.$$

#### Example: Interval and Ellipsoidal Robust Counterparts of Uncertain Linear Constraint with Independent Random Perturbations of Coefficients

Consider a linear constraint

$$l \leq \sum_{j=1}^{n} a_j x_j \leq u \tag{9.15}$$

and assume that the $a_j$ coefficients of the body of the constraint are uncertain and vary in intervals $a_j^{\mathrm{n}} \pm \sigma_j$. The worst-case_oriented model of uncertainty here is the interval one, and the corresponding robust counterpart of the constraint is given by the system of linear inequalities

$$
\begin{array}{rcll}
l & \leq & \sum_{j=1}^n a_j^{\mathrm{n}} x_j - \sum_{j=1}^n \sigma_j y_j, & \\
& & \sum_{j=1}^n a_j^{\mathrm{n}} x_j + \sum_{j=1}^n \sigma_j y_j & \leq u, \\
0 & \leq & x_j + y_j, & \\
0 & \leq & -x_j + y_j, & j = 1, \ldots, n.
\end{array} \tag{9.16}
$$

Now, assume that we have reasons to believe that the true values of the coefficients $a_j$ are obtained from their nominal values $a_j^{\mathrm{n}}$ by random perturbations, independent for different $j$ and symmetrically distributed in the segments *[-sigma_j,sigma_j]*. With this assumption, we are in the situation of item 3 and can replace the uncertain constraint (9.15) with its ellipsoidal robust counterpart

$$
\begin{array}{rcll}
l & \leq & \sum_{j=1}^n a_j^{\mathrm{n}} x_j - \Omega z, & \\
& & \sum_{j=1}^n a_j^{\mathrm{n}} x_j + \Omega z & \leq u, \\
0 & \leq & z - \sqrt{\sum_{j=1}^n \sigma_j^2 x_j^2}. &
\end{array} \tag{9.17}
$$

Please note that with the model of random perturbations, a vector $x$ satisfying (9.17) satisfies a realization of (9.15) with probability at least $1 - \exp\{\Omega^2/2\}$; for $\Omega = 6$. This probability is $\geq 1 - 1.5 \cdot 10^{-8}$, which for all practical purposes is the same as sayiong that $x$ satisfies all realizations of (9.15). On the other hand, the uncertainty set associated with (9.16) is the box

$$B = \left\{a = (a_1, \ldots, a_n)^T : a_j^{\mathrm{n}} - \sigma_j \leq a_j \leq a_j^{\mathrm{n}} + \sigma_j, j = 1, \ldots, n\right\},$$

while the uncertainty set associated with (9.17) is the ellipsoid

$$E(\Omega) = \left\{a = (a_1, \ldots, a_n)^T : \sum_{j=1}^n (a_j - a_j^{\mathrm{n}})^{\frac{2}{\sigma_j^2}} \leq \Omega^2\right\}.$$

For a moderate value of $\Omega$, say $\Omega = 6$, and $n \geq 40$, the ellipsoid $E(\Omega)$ in its diameter, typical linear sizes, volume, etc. is incomparably less than the box $B$, the difference becoming more dramatic the larger the dimension $n$ of the box and the ellipsoid. It follows that the ellipsoidal robust counterpart

(9.17) of the randomly perturbed uncertain constraint (9.15) is much less conservative than the interval robust counterpart (9.16), while ensuring basically the same "robustness guarantees". To illustrate this important point, consider the following numerical examples:

There are $n$ different assets on the market. The return on 1 invested in asset $j$ is a random variable distributed symmetrically in the segment $[\delta_j - \sigma_j, \delta_j + \sigma_j]$, and the returns on different assets are independent of each other. The problem is to distribute ' 1 among the assets in order to get the largest possible total return on the resulting portfolio.

A natural model of the problem is an uncertain linear optimization problem

$$\begin{array}{lrcll} \text{maximize} & & & \sum_{j=1}^{n} a_j x_j & \\ \text{subject to} & & & \sum_{j=1}^{n} x_j & = & 1, \\ & 0 & \leq & x_j, & & j = 1, \ldots, n. \end{array}$$

where $a_j$ are the uncertain returns of the assets. Both the nominal optimal solution (set all returns $a_j$ equal to their nominal values $\delta_j$) and the risk-neutral Stochastic Programming approach (maximize the expected total return) result in the same solution: Our money should be invested in the most promising asset(s) – the one(s) with the maximal nominal return. This solution, however, can be very unreliable if, as is typically the case in reality, the most promising asset has the largest volatility $\sigma$ and is in this sense the most risky. To reduce the risk, one can use the Robust Counterpart approach which results in the following optimization problems.

### The Interval Model of Uncertainty:

$$\begin{array}{lrcll} \text{maximize} & & & t & \\ \text{subject to} & 0 & \leq & -t + \sum_{j=1}^{n} (\delta_j - \sigma_j) x_j, & \\ & & & \sum_{j=1}^{n} x_j & = & 1, \\ & 0 & \leq & x_j, & & j = 1, \ldots, n \end{array} \tag{9.18}$$

and

### The ellipsoidal Model of Uncertainty:}

$$\begin{array}{lrcll} \text{maximize} & & & t & \\ \text{subject to} & 0 & \leq & -t + \sum_{j=1}^{n} (\delta_j) x_j - \Omega z, & \\ & 0 & \leq & z - \sqrt{\sum_{j=1}^{n} \sigma_j^2 x_j^2}, & \\ & & & \sum_{j=1}^{n} x_j & = & 1, \\ & 0 & \leq & x_j, & & j = 1, \ldots, n. \end{array} \tag{9.19}$$

Note that the problem (9.19) is essentially the risk-averted portfolio model proposed in mid-50's by Markowitz.

The solution of (9.18) is evident — our '1 should be invested in the asset(s) with the largest possible *guaranteed* return $\delta_j - \sigma_j$. In contrast to this very conservative policy (which in reality prescribes to keep the initial capital in a bank or in the most reliable, and thus low profit, assets), the optimal solution to (9.19) prescribes a quite reasonable diversification of investments which allows to get much better total return than (9.18) with basically zero risk[2]. To illustrate this, assume that there are $n = 300$ assets with the nominal returns (per year) varying from 1.04 (bank savings) to 2.00:

$$\delta_j = 1.04 + 0.96 \frac{j-1}{n-1}, \quad j = 1, 2, \ldots, n = 300$$

[2] Recall that in our discussion we have assumed the returns on different assets to be independent of each other. In reality, this is not so and this is why diversification of investments, although reducing the risk, never eliminates it completely

and volatilities varying from 0 for the bank savings to 1.2 for the most promising asset:

$$\sigma_j = 1.152\frac{j-1}{n-1}, \quad j = 1, \ldots, n = 300.$$

In Listing 9.4 a MATLAB script which builds the associated problem (9.19), solves it via the **MOSEK** optimization toolbox, displays the resulting robust optimal value of the total return and the distribution of investments, and finally runs 10,000 simulations to get the distribution of the total return on the resulting portfolio (in these simulations, the returns on all assets are uniformly distributed in the corresponding intervals) is presented.

Listing 9.4: Script that implements problem (9.19).

```
function rlo2(n, Omega, draw)

n = str2num(n)
Omega = str2num(Omega)
draw

% Set nominal returns and volatilities
delta = (0.96/(n-1))*[0:1:n-1]+1.04;
sigma = (1.152/(n-1))*[0:1:n-1];

% Set mosekopt description of the problem
prob.c = -[1;zeros(2*n+1,1)];
A      = [-1,ones(1,n)+delta,-Omega,zeros(1,n);zeros(n+1,2*n+2)];
for j=1:n,
    % Body of the constraint y(j) - sigma(j)*x(j) = 0:
    A(j+1,j+1)   = -sigma(j);
    A(j+1,2+n+j) = 1;
end;
A(n+2,2:n+1)        = ones(1,n);
prob.a             = sparse(A);
prob.blc           = [zeros(n+1,1);1];
prob.buc           = [inf;zeros(n,1);1];
prob.blx           = [-inf;zeros(n,1);0;zeros(n,1)];
prob.bux           = inf*ones(2*n+2,1);
prob.cones         = cell(1,1);
prob.cones{1}.type = 'MSK_CT_QUAD';
prob.cones{1}.sub  = [n+2;[n+3:1:2*n+2]'];

% Run mosekopt
[r,res]=mosekopt('minimize echo(1)',prob);

if draw == true
    % Display the solution
    xx = res.sol.itr.xx;
    t  = xx(1);

    disp(sprintf('Robust optimal value: %5.4f',t));
    x = max(xx(2:1+n),zeros(n,1));
    plot([1:1:n],x,'-m');
    grid on;

    disp('Press <Enter> to run simulations');
    pause

    % Run simulations

    Nsim = 10000;
    out  = zeros(Nsim,1);
    for i=1:Nsim,
        returns  = delta+(2*rand(1,n)-1).*sigma;
        out(i)   = returns*x;
```

```
    end;
    disp(sprintf('Actual returns over %d simulations:',Nsim));
    disp(sprintf('Min=%5.4f Mean=%5.4f Max=%5.4f StD=%5.2f',...
                 min(out),mean(out),max(out),std(out)));
    hist(out);
end
```

Here are the results displayed by the script:

Listing 9.5: Output of script *rlo2.m*.

```
Robust optimal value: 1.3428
Actual returns over 10000 simulations:
Min=1.5724 Mean=1.6965 Max=1.8245 StD= 0.03
```

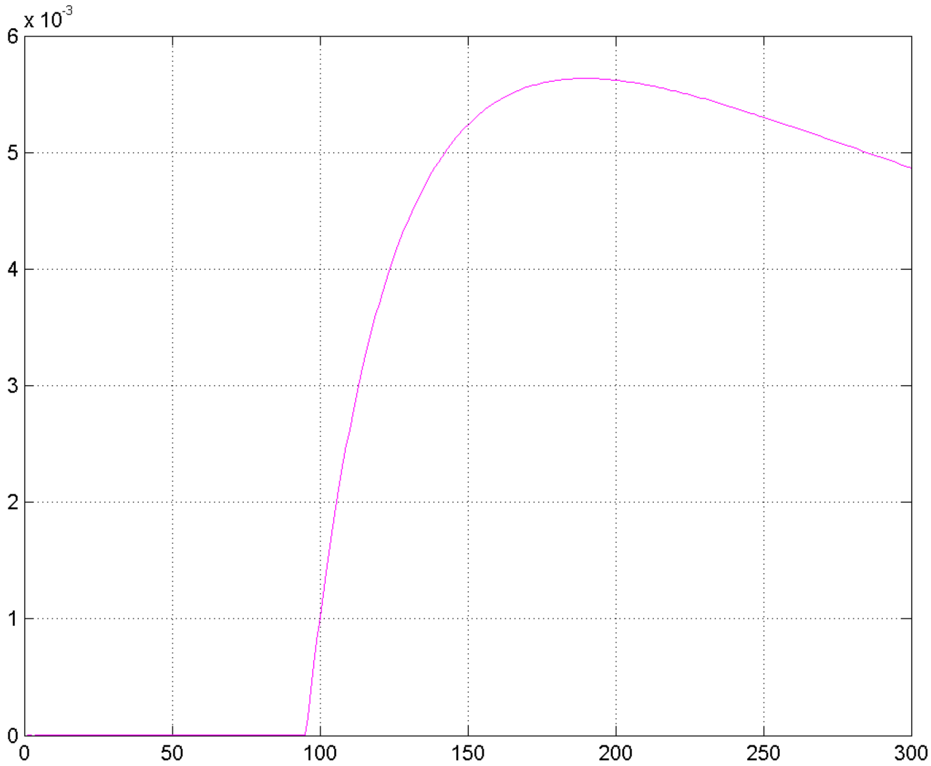


Fig. 9.1: Distribution of investments among the assets in the optimal solution of.

Please note that with our set-up there is exactly one asset with guaranteed return greater than 1 – asset # 1 (bank savings, return 1.04, zero volatility). Consequently, the interval robust counterpart (9.18) prescribes to put our ' #1 in the bank, thus getting a 4% profit. In contrast to this, the diversified portfolio given by the optimal solution of (9.19) never yields profit less than 57.2%, and yields at average a 69.67% profit with pretty low (0.03) standard deviation. We see that in favorable circumstances the ellipsoidal robust counterpart of an uncertain linear program indeed is less conservative than, although basically as reliable as, the interval robust counterpart.

Finally, let us compare our results with those given by the nominal optimal solution. The latter prescribes to invest everything we have in the most promising asset (in our example this is the asset # 300 with a nominal return of 2.00 and volatility of 1.152). Assuming that the actual return is uniformly distributed in the corresponding interval and running 10,000 simulations, we get the following results:

```
Nominal optimal value: 2.0000
Actual returns over 10000 simulations:
Min=0.8483 Mean=1.9918 Max=3.1519 StD= 0.66
```

We see that the nominal solution results in a portfolio which is much more risky, although better at average, than the portfolio given by the robust solution.

#### Combined Interval-Ellipsoidal Robust Counterpart

We have considered the case when the coefficients $a_j$ of uncertain linear constraint (9.15) are affected by uncorrelated random perturbations symmetrically distributed in given intervals $[-\sigma_j, \sigma_j]$, and we have discussed two ways to model the uncertainty:

- The interval uncertainty model (the uncertainty set $\mathcal{U}$ is the box $B$), where we ignore the stochastic nature of the perturbations and their independence. This model yields the Interval Robust Counterpart (9.16);
- The ellipsoidal uncertainty model ($\mathcal{U}$ is the ellipsoid $E(\Omega)$), which takes into account the stochastic nature of data perturbations and yields the Ellipsoidal Robust Counterpart (9.17).

Please note that although for large $n$ the ellipsoid $E(\Omega)$ in its diameter, volume and average linear sizes is incomparably smaller than the box $B$, in the case of $\Omega > 1$ the ellipsoid $E(\Omega)$ in certain directions goes beyond the box. E.g. the ellipsoid $E(6)$, although much more narrow than $B$ in most of the directions, is 6 times wider than $B$ in the directions of the coordinate axes. Intuition says that it hardly makes sense to keep in the uncertainty set realizations of the data which are outside of $B$ and thus forbidden by our model of perturbations, so in the situation under consideration the intersection of $E(\Omega)$ and $B$ is a better model of the uncertainty set than the ellipsoid $E(\Omega)$ itself. What happens when the model of the uncertainty set is the *combined interval-ellipsoidal* uncertainty $\mathcal{U}(\Omega) = E(\Omega) \cap B$?

First, it turns out that the RC of (9.15) corresponding to the uncertainty set $\mathcal{U}(\Omega)$ is still given by a system of linear and conic quadratic inequalities, specifically the system

$$\begin{array}{rclcl} l & \leq & \sum_{j=1}^n a_j^{\mathrm{n}} x_j - \sum_{j=1}^n \sigma_j y_j - \Omega\sqrt{\sum_{j=1}^n \sigma_j^2 u_j^2}, & & \\ & & \sum_{j=1}^n a_j^{\mathrm{n}} x_j + \sum_{j=1}^n \sigma_j z_j + \Omega\sqrt{\sum_{j=1}^n \sigma_j^2 v_j^2} & \leq & u, \\ -y_j & \leq & x_j - u_j & \leq & y_j, j = 1, \ldots, n, \\ -z_j & \leq & x_j - v_j & \leq & z_j, j = 1, \ldots, n. \end{array} \tag{9.20}$$

Second, it turns out that our intuition is correct: As a model of uncertainty, $U(\Omega)$ is as reliable as the ellipsoid $E(\Omega)$. Specifically, if $x$ can be extended to a feasible solution of (9.20), then the probability for $x$ to satisfy a realization of (9.15) is $\geq 1 - \exp\{-\Omega^2/2\}$.

The conclusion is that if we have reasons to assume that the perturbations of uncertain coefficients in a constraint of an uncertain linear optimization problem are (a) random, (b) independent of each other, and (c) symmetrically distributed in given intervals, then it makes sense to associate with this constraint an interval-ellipsoidal model of uncertainty and use a system of linear and conic quadratic inequalities (9.20). Please note that when building the robust counterpart of an uncertain linear optimization problem, one can use different models of the uncertainty (e.g., interval, ellipsoidal, combined interval-ellipsoidal) for different uncertain constraints within the same problem.

## 9.2 Geometric (posynomial) Optimization

### 9.2.1 Problem Definition

A *geometric optimization* problem can be stated as follows

$$\begin{array}{lccl} \text{minimize} & \sum_{k \in J_0} c_k \prod_{j=0}^{n-1} t_j^{a_{kj}} & & \\ \text{subject to} & \sum_{k \in J_i} c_k \prod_{j=0}^{n-1} t_j^{a_{kj}} & \leq & 1, \quad i = 1, \ldots, m, \\ & t > 0, & & \end{array} \tag{9.21}$$

where it is assumed that

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

and if $i \neq j$, then

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

Hence, $A$ is a $T \times n$ matrix and $c$ is a vector of length $T$. Given $c_k > 0$ then

$$c_k \prod_{j=0}^{n-1} t_j^{a_{kj}}$$

is called a *monomial*. A sum of monomials i.e.

$$\sum_{k \in J_i} c_k \prod_{j=0}^{n-1} t_j^{a_{kj}}$$

is called a *posynomial* posynomial In general, problem (9.21) is very hard to solve. However, the posynomial case where it is required that

$$c > 0$$

is relatively easy. The reason is that using a simple variable transformation a convex optimization problem can be obtained. Indeed using the variable transformation

$$t_j = e^{x_j}$$

we obtain the problem

$$\begin{array}{lll} \text{minimize} & \sum_{k \in J_0} c_k e^{\sum_{j=0}^{n-1} a_{kj} x_j} & \\ \text{subject to} & \sum_{k \in J_i} c_k e^{\sum_{j=0}^{n-1} a_{kj} x_j} & \leq \quad 1, \quad i = 1, \ldots, m, \end{array} \tag{9.22}$$

which is a convex optimization problem that can be solved using **MOSEK**. We will call

$$c_t e^{\left\{\sum_{j=0}^{n-1} a_{tj} x_j\right\}} = e^{\left\{\log(c_t) + \sum_{j=0}^{n-1} a_{tj} x_j\right\}}$$

a *term* and hence the number of terms is $T$.

As stated, problem (9.22) is non-separable. However, using

$$v_t = \log(c_t) + \sum_{j=0}^{n-1} a_{tj} x_j$$

we obtain the separable problem

$$\begin{array}{lrcll} \text{minimize} & \sum_{t \in J_0} e^{v_t} & & & \\ \text{subject to} & \sum_{t \in J_i} e^{v_t} & \leq & 1, & i = 1, \ldots, m, \\ & \sum_{j=0}^{n-1} a_{tj} x_j - v_t & = & -\log(c_t), & t = 0, \ldots, T, \end{array}$$

which is a separable convex optimization problem.

A warning about this approach is that the exponential function $e^x$ is only numerically well-defined for values of $x$ in a small interval around 0 since $e^x$ grows very rapidly as $x$ becomes larger. Therefore numerical problems may arise when solving the problem on this form.

### Applications

A large number of practical applications, particularly in electrical circuit design, can be cast as a geometric optimization problem. We will not review these applications here but rather refer the reader to *[BKVH04]* and the references therein.

#### Further Information

More information about geometric optimization problems is located in [BSS93], [BP76], [BKVH04].

#### Modeling tricks

A lot of tricks that can be used for modeling posynomial optimization problems are described in [BKVH04]. Therefore, in this section we cover only one important case.

#### Equalities

In general, equalities are not allowed in (9.21), i.e.

$$\sum_{k \in J_i} c_k \prod_{j=0}^{n-1} t_j^{a_{kj}} = 1$$

is not allowed. However, a monomial equality is not a problem. Indeed consider the example

$$xyz^{-1} = 1$$

of a monomial equality. The equality is identical to

$$1 \leq xyz^{-1} \leq 1$$

which in turn is identical to the two inequalities

$$\begin{array}{lclcl} xyz^{-1} & & & \leq & 1, \\ \frac{1}{xyz^{-1}} & = & x^{-1}y^{-1}z & \leq & 1. \end{array}$$

Hence, it is possible to model a monomial equality using two inequalities.

### 9.2.2 Problematic Formulations

Certain formulations of geometric optimization problems may cause problems for the algorithms implemented in **MOSEK**. Basically there are two kinds of problems that may occur:

- The solution vector is finite, but an optimal objective value can only be a approximated.
- The optimal objective value is finite but implies that a variable in the solution is infinite.

#### Finite Unattainable Solution

The following problem illustrates an unattainable solution:

$$\begin{array}{lccc} \text{minimize} & x^2 y & & \\ \text{subject to} & xy & \leq & 1, \\ & x, y > 0. & & \end{array}$$

Clearly, the optimal objective value is 0 but because of the constraint the $x, y > 0$ constraint this value can never be attained: To see why this is a problem, remember that **MOSEK** substitutes $x = e^{t_x}$ and $y = e^{t_y}$ and solves the problem as

$$\begin{array}{lccc} \text{minimize} & e^{2t_x} e^{t_y} & & \\ \text{subject to} & e^{t_x} e^{t_y} & \leq & 1, \\ & t_x, t_y \in \mathbb{R}. & & \end{array}$$

The optimal solution implies that $t_x = -\infty$ or $t_y = -\infty$, and thus it is unattainable.

Now, the issue should be clear: If a variable $x$ appears only with nonnegative exponents, then fixing $x = 0$ will minimize all terms in which it appears — but such a solution cannot be attained.

### Infinite Solution

A similar problem will occur if a finite optimal objective value requires a variable to be infinite. This can be illustrated by the following example:

$$\begin{array}{lll} \text{minimize} & x^{-2} & \\ \text{subject to} & x^{-1} & \leq \quad 1, \\ & x > 0, & \end{array}$$

which is a valid geometric programming problem. In this case the optimal objective is 0, but this requires $x = \infty$, which is unattainable.

Again, this specific case will appear if a variable $x$ appears only with negative exponents in the problem, implying that each term in which it appears can be minimized for $x \to \infty$.

## 9.2.3 An Example

Consider the example

$$\begin{array}{lccc} \text{minimize} & x^{-1}y & & \\ \text{subject to} & x^2 y^{-\frac{1}{2}} + 3y^{\frac{1}{2}} z^{-1} & \leq & 1, \\ & xy^{-1} & = & z^2, \\ & -x & \leq & -\frac{1}{10}, \\ & x & \leq & 3, \\ & x, y, z > 0, & & \end{array}$$

which is not a geometric optimization problem. However, using the obvious transformations we obtain the problem

$$\begin{array}{lccc} \text{minimize} & x^{-1}y & & \\ \text{subject to} & x^2 y^{-\frac{1}{2}} + 3y^{\frac{1}{2}} z^{-1} & \leq & 1, \\ & xy^{-1}z^{-2} & \leq & 1, \\ & x^{-1}yz^2 & \leq & 1, \\ & \frac{1}{10}x^{-1} & \leq & 1, \\ & \frac{1}{3}x & \leq & 1, \\ & x, y, z > 0, & & \end{array} \tag{9.23}$$

which is a geometric optimization problem.

## 9.2.4 Solving the Example

The problem (9.23) can be defined and solved in the **MOSEK** toolbox as shown in Listing 9.6.

Listing 9.6: Script implementing problem (9.23).

```
function go2()
c     = [1 1 3 1 1 0.1 1/3]';
a     = sparse([[-1  1  0];
                [2 -0.5 0];
                [0 0.5 -1];
                [1 -1 -2];
                [-1 1 2];
                [-1 0 0];
                [1 0 0]]);

map   = [0 1 1 2 3 4 5]';
[res] = mskgpopt(c,a,map);

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

```
fprintf('\n\n');

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

% Add appropriate terms together.
f = sparse(map+1,1:7,ones(size(map)))*v;

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

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

### 9.2.5 Exporting to a File

It's possible to write a geometric optimization problem to a file with the command:

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

This file format is compatible with the *mskenopt* command line tool. See the **MOSEK** Tools User's manual for details on *mskenopt*. This file format can be useful for sending debug information to **MOSEK** or for testing. It's also possible to read the above format with the command:

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

CHAPTER

TEN

# MANAGING I/O

The main purpose of this chapter is to give an overview on the logging and I/O features provided by the **MOSEK** package.

- Sec. 10.1 contains information about the log streams provided by **MOSEK**.
- File I/O is discussed in Sec. 10.2.
- How to tune the logging verbosity is the topic of Sec. 10.3.

## 10.1 Stream I/O

**MOSEK** execution produces a certain amount of loggging at environment and task level. This means that the logging from each environement and task can be isolated from the others.

The log messages are partitioned in three streams:

- *messages*
- *warnings*
- *errors*

These streams are aggregated in the *log* stream.

## 10.2 File I/O

**MOSEK** supports a range of problem and solution formats listed in Sec. 18. One such format is **MOSEK**'s native binary *Task format* which supports all features that **MOSEK** supports.

The file format used in I/O operations is deduced from extension - as in `problemname.task` - unless the parameter *MSK_IPAR_WRITE_DATA_FORMAT* is specified to something else. Problem files with an additional `.gz` extension - as in `problemname.task.gz` - are moreover assumed to use GZIP compression, and are automatically compressed, respectively decompressed, when written or read.

### Example

If something is wrong with a problem or a solution, one option is to output the problem to the human-readable *OPF format* and inspect it by hand. For instance, one may use the *mosekopt* function to write the problem to a file immediately before optimizing it:

```
% Write the data defined by prob to an OPF file
% named datafile.mps
mosekopt('write(datafile.opf)',prob);
```

This will write the problem in `prob` to the file `datafile.opf`.

When using MATLAB-like functions, as for instance *linprog*, control parameters can be set using the `options` structure, for example,

```
options.Write = 'test.opf';
linprog(f,A,b,B,c,l,u,x0,options);
```

which will also write the problem to an `opf`-formatted file before optimizing.

## 10.3 Verbosity

The logging verbosity can be controlled by setting the relevant paramenters, as for instance

- *MSK_IPAR_LOG*,
- *MSK_IPAR_LOG_INTPNT*,
- *MSK_IPAR_LOG_MIO*,
- *MSK_IPAR_LOG_CUT_SECOND_OPT*,
- *MSK_IPAR_LOG_SIM*, and
- *MSK_IPAR_LOG_SIM_MINOR*.

Each parameter control the output level of a specific functionality or algorithm. The main switch is *MSK_IPAR_LOG* which affect the whole output. The actual log level for a specific functionality is determined as the minimum between *MSK_IPAR_LOG* and the relevant parameter. For instance, the log level for the output produce by the interior-point algorithm is tuned by the *MSK_IPAR_LOG_INTPNT*: the actual log level is defined by the minimum between *MSK_IPAR_LOG* and *MSK_IPAR_LOG_INTPNT*.

Tuning the solver verbosity may require adjusting several parameters. It must be noticed that verbose logging is supposed to be of interest during debugging and tuning, and it is consider the default setting. When output is no more of interest, user can easily disable using *MSK_IPAR_LOG*.

Moreover, it must be understood that larger values of *MSK_IPAR_LOG* do not necessarily result in an increased output.

By default **MOSEK** will reduce the amount of log information after the first optimization on a given task. To get full log output on subsequent optimizations set *MSK_IPAR_LOG_CUT_SECOND_OPT* to zero.

CHAPTER

# ELEVEN

# THE OPTIMIZERS FOR CONTINUOUS PROBLEMS

The most essential part of **MOSEK** are the optimizers. This chapter describes the optimizers for the class of *continuous problems* without integer variables, that is:

- linear problems,
- conic problems (quadratic and semidefinite),
- general convex problems.

**MOSEK** offers an interior-point optimizer for each class of problems and also a simplex optimizer for linear problems. 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.

## 11.1 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 *[AGMX96]*.

It is possible to fine-tune the behavior of the presolve or to turn it off entirely. If presolve consumes too much time or memory compared to the reduction in problem size gained it may be disabled. This is done by setting the parameter *MSK_IPAR_PRESOLVE_USE* to *"MSK_PRESOLVE_MODE_OFF"*. The two most time-consuming steps of the presolve are

- the eliminator, and
- the linear dependency check.

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

### Numerical issues in the presolve

During the presolve the problem is reformulated so that it hopefully solves faster. However, in rare cases the presolved problem may be harder to solve then the original problem. The presolve may also be infeasible although the 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

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

$y$ is an implied free variable that can be substituted out of the problem, if deemed worthwhile. If the eliminator consumes too much time or memory compared to the reduction in problem size gained it may be disabled. This can be done 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

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

contain exactly one linear dependency. This implies that one of the constraints can be dropped without changing the set of feasible solutions. Removing linear dependencies is in general a good idea since it reduces the size of the problem. Moreover, the linear dependencies are likely to introduce numerical problems in the optimization phase. It is best 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 modelling 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 quadratic 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.

## 11.2 Using Multiple Threads in an Optimizer

### Multithreading in interior-point optimizers

The interior-point optimizers in **MOSEK** have been parallelized. This means that if you solve linear, quadratic, conic, or general convex optimization problem using the interior-point optimizer, you can take advantage of multiple CPU's. By default **MOSEK** will automatically select the number of threads to be employed when solving the problem. However, the maximum number of threads employed can be changed by setting the parameter *MSK_IPAR_NUM_THREADS*. This should never exceed the number of cores on the computer.

The speed-up obtained when using multiple threads is highly problem and hardware dependent, and consequently, it is advisable to compare single threaded and multi threaded performance for the given problem type to determine the optimal settings. For small problems, using multiple threads is not be worthwhile and may even be counter productive because of the additional coordination overhead. Therefore, it may be advantageous to disable multithreading using the parameter *MSK_IPAR_INTPNT_MULTI_THREAD*.

The interior-point optimizer parallelizes big tasks such linear algebra computations.

### Thread Safety

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

#### Determinism

The optimizers are run-to-run deterministic which means if a problem is solved twice on the same computer using the same parameter setting and exactly the same input then exactly the same results is obtained. One restriction is that no time limits must be imposed because the time taken to perform an operation on a computer is dependent on many factors such as the current workload.

## 11.3 Linear Optimization

### 11.3.1 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.

### 11.3.2 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

$$\begin{array}{lccl} \text{minimize} & c^T x & & \\ \text{subject to} & Ax & = & b, \\ & x \geq 0. & & \end{array} \tag{11.1}$$

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 (11.1) has an optimal solution, is primal infeasible or is dual infeasible, the optimization algorithm must deal with all three situations. This is the reason why **MOSEK** solves the so-called homogeneous model

$$\begin{array}{rcl} Ax - b\tau & = & 0, \\ A^T y + s - c\tau & = & 0, \\ -c^T x + b^T y - \kappa & = & 0, \\ x, s, \tau, \kappa & \geq & 0, \end{array} \tag{11.2}$$

where $y$ and $s$ correspond to the dual variables in (11.1), and $\tau$ and $\kappa$ are two additional scalar variables. Note that the homogeneous model (11.2) always has solution since

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

is a solution, although not a very interesting one. Any solution

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

to the homogeneous model (11.2) satisfies

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

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

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

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

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. 16.1 for the mathematical background on duality and optimality).

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

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

This implies that at least one of

$$c^T x^* < 0 \tag{11.3}$$

or

$$b^T y^* > 0 \tag{11.4}$$

is satisfied. If (11.3) is satisfied then $x^*$ is a certificate of dual infeasibility, whereas if (11.4) 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

$$\begin{array}{rcl} \left\| A\frac{x^k}{\tau^k} - b \right\|_\infty & \leq & \epsilon_p (1 + \|b\|_\infty), \\ \left\| A^T\frac{y^k}{\tau^k} + \frac{s^k}{\tau^k} - c \right\|_\infty & \leq & \epsilon_d (1 + \|c\|_\infty), \text{ and} \\ \min\left(\frac{(x^k)^T s^k}{(\tau^k)^2}, \left|\frac{c^T x^k}{\tau^k} - \frac{b^T y^k}{\tau^k}\right|\right) & \leq & \epsilon_g \max\left(1, \frac{\min(|c^T x^k|, |b^T y^k|)}{\tau^k}\right), \end{array} \tag{11.5}$$

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 (11.5) 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

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

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

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

and define

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

It is easy to verify that

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

which shows $\bar{x}$ is an approximate certificate of dual infeasibility, where $\varepsilon_i$ 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 and near optimality

It is possible to adjust the tolerances $\varepsilon_p$, $\varepsilon_d$, $\varepsilon_g$ and $\varepsilon_i$ using parameters; see table for details.

Table 11.1: Parameters employed in termination criterion

| ToleranceParameter | name |
|---|---|
| $\varepsilon_p$ | *MSK_DPAR_INTPNT_TOL_PFEAS* |
| $\varepsilon_d$ | *MSK_DPAR_INTPNT_TOL_DFEAS* |
| $\varepsilon_g$ | *MSK_DPAR_INTPNT_TOL_REL_GAP* |
| $\varepsilon_i$ | *MSK_DPAR_INTPNT_TOL_INFEAS* |

The default values of the termination tolerances are chosen such that for a majority of problems appearing in practice it is not possible to achieve much better accuracy. Therefore, tightening the tolerances usually is not worthwhile. However, an inspection of (11.5) 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, $\varepsilon_p$, $\varepsilon_d$, $\varepsilon_g$ and $\varepsilon_i$, have to be relaxed together to achieve an effect.

In some cases the interior-point method terminates having found a solution not too far from meeting the optimality condition (11.5). A solution is defined as *near optimal* if scaling the termination tolerances $\varepsilon_p$, $\varepsilon_d$, $\varepsilon_g$ and $\varepsilon_g$ by the same factor $\varepsilon_n \in [1.0, +\infty]$ makes the condition (11.5) satisfied. A near optimal solution is therefore of lower quality but still potentially valuable. If for instance the solver stalls, i.e. it can make no more significant progress towards the optimal solution, a near optimal solution could be available and be good enough for the user. Near infeasibility certificates are defined similarly. The value of $\varepsilon_n$ can be adjusted with the parameter *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL* .

The basis identification discussed in Sec. 11.3.2 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:

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

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

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

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
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 tells that the optimizer chose to solve the dual problem rather than the primal problem. The next line displays the problem dimensions as seen by the optimizer, and the `Factor...` lines show various statistics. This is followed by the iteration log.

Using the same notation as in Sec. 11.3.2 the columns of the iteration log have the following meaning:

- `ITE`: Iteration index $k$.
- `PFEAS`: $\left\|Ax^k - b\tau^k\right\|_\infty$ . The numbers in this column should converge monotonically towards zero but may stall at low level due to rounding errors.
- `DFEAS`: $\left\|A^T y^k + s^k - c\tau^k\right\|_\infty$ . The numbers in this column should converge monotonically 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.

### 11.3.3 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. 11.3.1 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. 16.1 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
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).

## 11.4 Conic Optimization

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

### 11.4.1 The Interior-point optimizer

#### 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:

$$\begin{array}{lccc} \text{minimize} & c^T x & & \\ \text{subject to} & Ax & = & b, \\ & x \in \mathcal{K} & & \end{array} \tag{11.6}$$

where $K$ is a convex cone. The corresponding dual problem is

$$\begin{array}{lccc} \text{maximize} & b^T y & & \\ \text{subject to} & A^T y + s & = & c, \\ & x \in \mathcal{K}^* & & \end{array} \tag{11.7}$$

where $\mathcal{K}^*$ is the dual cone of $\mathcal{K}$. See Sec. 16.2 for definitions.

Since it is not known beforehand whether problem (11.6) 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

$$\begin{array}{rcl} Ax - b\tau & = & 0, \\ A^T y + s - c\tau & = & 0, \\ -c^T x + b^T y - \kappa & = & 0, \\ x & \in & \mathcal{K}, \\ s & \in & \mathcal{K}^*, \\ \tau, \kappa & \geq & 0, \end{array} \tag{11.8}$$

where $y$ and $s$ correspond to the dual variables in (11.6), and $\tau$ and $\kappa$ are two additional scalar variables. Note that the homogeneous model (11.8) 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 (11.8) 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 $\tau^*$ and $\kappa^*$ is zero.

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

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

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

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

is a primal-dual optimal solution.

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

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

This implies that at least one of

$$c^T x^* < 0 \tag{11.9}$$

or

$$b^T y^* > 0 \tag{11.10}$$

holds. If (11.9) is satisfied, then $x^*$ is a certificate of dual infeasibility, whereas if (11.10) 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]*.

### 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:

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

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

### Optimal Case

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

$$\left\| A\frac{x^k}{\tau^k} - b \right\|_\infty \leq \varepsilon_p(1 + \|b\|_\infty). \tag{11.11}$$

This shows the violations in the primal equality constraints for the solution $x^k/\tau^k$ is small compared to the size of $b$ given $\varepsilon_p$ 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

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

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 $\varepsilon_i$ controls the quality of the approximation.

### Primal Infeasiblity Certificate

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

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

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

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

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

### Adjusting optimality criteria and near optimality

It is possible to adjust the tolerances $\varepsilon_p$, $\varepsilon_d$, $\varepsilon_g$ and $\varepsilon_i$ using parameters; see table for details.

Table 11.2: Parameters employed in termination criterion

| ToleranceParameter | name |
|---|---|
| $\varepsilon_p$ | *MSK_DPAR_INTPNT_CO_TOL_PFEAS* |
| $\varepsilon_d$ | *MSK_DPAR_INTPNT_CO_TOL_DFEAS* |
| $\varepsilon_g$ | *MSK_DPAR_INTPNT_CO_TOL_REL_GAP* |
| $\varepsilon_i$ | *MSK_DPAR_INTPNT_CO_TOL_INFEAS* |

The default values of the termination tolerances are chosen such that for a majority of problems appearing in practice it is not possible to achieve much better accuracy. Therefore, tightening the tolerances usually is not worthwhile. However, an inspection of (11.11) 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, $\varepsilon_p$, $\varepsilon_d$, $\varepsilon_g$ and $\varepsilon_i$, have to be relaxed together to achieve an effect.

In some cases the interior-point method terminates having found a solution not too far from meeting the optimality condition (11.11). A solution is defined as *near optimal* if scaling the termination tolerances $\varepsilon_p$, $\varepsilon_d$, $\varepsilon_g$ and $\varepsilon_g$ by the same factor $\varepsilon_n \in [1.0, +\infty]$ makes the condition (11.11) satisfied. A near optimal solution is therefore of lower quality but still potentially valuable. If for instance the solver stalls, i.e. it can make no more significant progress towards the optimal solution, a near optimal solution could be available and be good enough for the user. Near infeasibility certificates are defined similarly. The value of $\varepsilon_n$ can be adjusted with the parameter *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL* .

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

### 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                  : 1.70e+01
ITE PFEAS    DFEAS    GFEAS    PRSTATUS   POBJ              DOBJ              MU       TIME
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 tells that the optimizer chose to solve the dual problem rather than the primal problem. The next line displays the problem dimensions as seen by the optimizer, and the `Factor...` lines show various statistics. This is followed by the iteration log.

Using the same notation as in Sec. 11.4.1 the columns of the iteration log have the following meaning:

- `ITE`: Iteration index $k$.
- `PFEAS`: $\left\|Ax^k - b\tau^k\right\|_\infty$ . The numbers in this column should converge monotonically towards zero but may stall at low level due to rounding errors.
- `DFEAS`: $\left\|A^T y^k + s^k - c\tau^k\right\|_\infty$ . The numbers in this column should converge monotonically 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).

## 11.5 Nonlinear Convex Optimization

### 11.5.1 The Interior-point Optimizer

For general convex optimization problems an interior-point type optimizer is available. The interior-point optimizer is an implementation of the homogeneous and self-dual algorithm. For a detailed description of the algorithm, please see *[AY98]*, *[AY99]*.

#### The Convexity Requirement

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

The following parameters are available to control the convexity check:

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

### The Differentiability Requirement

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

For instance, the function

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

is differentiable everywhere whereas the function

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

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

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

$$x \geq 0$$

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

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

$$f(x) = \frac{1}{x} + \frac{1}{1-x}. \tag{11.12}$$

In this case the bounds

$$0 \leq x \leq 1$$

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

For efficiency reasons it may be better to reformulate the problem than to force **MOSEK** to observe ranged bounds strictly. For instance, (11.12) can be reformulated as follows

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

### Interior-point Termination Criteria

The parameters controlling when the general convex interior-point optimizer terminates are shown in Table 11.3.

Table 11.3: Parameters employed in termination criteria.

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

CHAPTER

TWELVE

# THE OPTIMIZER FOR MIXED-INTEGER PROBLEMS

A problem is a mixed-integer optimization problem when one or more of the variables are constrained to be integer valued. Readers unfamiliar with integer optimization are recommended to consult some relevant literature, e.g. the book *[Wol98]* by Wolsey.

## 12.1 The Mixed-integer Optimizer Overview

**MOSEK** can solve mixed-integer

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

problems, at least as long as they do not contain both quadratic objective or constraints and conic constraints at the same time. The mixed-integer optimizer is specialized for solving linear and conic optimization problems. Pure quadratic and quadratically constrained problems are automatically converted to conic form.

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. If a time limit is set then this may not be case since the time taken to solve a problem is not deterministic. The mixed-integer optimizer is parallelized i.e. it can exploit multiple cores during the optimization.

The solution process can be split into these phases:

1. **Presolve:** See Sec. 11.1.
2. **Cut generation:** Valid inequalities (cuts) are added to improve the lower bound.
3. **Heuristic:** Using heuristics the optimizer tries to guess a good feasible solution. Heuristics can be controlled by the parameter `MSK_IPAR_MIO_HEURISTIC_LEVEL` .
4. **Search:** The optimal solution is located by branching on integer variables.

## 12.2 Relaxations and bounds

It is important to understand that, in a worst-case scenario, the time required to solve integer optimization problems grows exponentially with the size of the problem (solving mixed-integer problems is NP-hard). For instance, a problem with $n$ binary variables, may require time proportional to $2^n$ . The value of $2^n$ is huge even for moderate values of $n$.

In practice this implies that the focus should be on computing a near-optimal solution quickly rather than on locating an optimal solution. Even if the problem is only solved approximately, it is important to know how far the approximate solution is from an optimal one. In order to say something about the quality of an approximate solution the concept of *relaxation* is important.

Consider for example a mixed-integer optimization problem

$$\begin{array}{rclccl} z^* & = & \text{minimize} & c^T x & & \\ & & \text{subject to} & Ax & = & b, \\ & & & x \geq 0 & & \\ & & & x_j \in \mathbb{Z}, & & \forall j \in \mathcal{J}. \end{array} \tag{12.1}$$

It has the continuous relaxation

$$\begin{array}{rclccl} \underline{z} & = & \text{minimize} & c^T x & & \\ & & \text{subject to} & Ax & = & b, \\ & & & x \geq 0 & & \end{array} \tag{12.2}$$

obtained simply by ignoring the integrality restrictions. The relaxation is a continuous problem, and therefore much faster to solve to optimality with a linear (or, in the general case, conic) optimizer. We call the optimal value $\underline{z}$ the *objective bound*. The objective bound $\underline{z}$ normally increases during the solution search process when the continuous relaxation is gradually refined.

Moreover, if $\hat{x}$ is any feasible solution to (12.1) and

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

then

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

These two inequalities allow us to estimate the quality of the integer solution: it is no further away from the optimum than $\bar{z} - \underline{z}$ in terms of the objective value. Whenever a mixed-integer problem is solved **MOSEK** reports this lower bound so that the quality of the reported solution can be evaluated.

## 12.3 Termination Criterion

In general, it is time consuming to find an exact feasible and optimal solution to an integer optimization problem, though in many practical cases it may be possible to find a sufficiently good solution. The issue of terminating the mixed-integer optimizer is rather delicate and the user has numerous possibilities of influencing it with various parameters. The mixed-integer optimizer employs a relaxed feasibility and optimality criterion to determine when a satisfactory solution is located.

A candidate solution that is feasible for the continuous relaxation is said to be an *integer feasible solution* if the criterion

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

is satisfied, meaning that $x_j$ is at most $\delta_1$ from the nearest integer.

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

$$\bar{z} - \underline{z} \leq \max(\delta_2, \delta_3 \max(10^{-10}, |\bar{z}|))$$

is satisfied. If this is the case, the integer optimizer terminates and reports the integer feasible solution as an optimal solution. If an optimal solution cannot be located after the time specified by the parameter *MSK_DPAR_MIO_DISABLE_TERM_TIME* (in seconds), it may be advantageous to relax the termination criteria, and they become replaced with

$$\bar{z} - \underline{z} \leq \max(\delta_4, \delta_5 \max(10^{-10}, |\bar{z}|)).$$

Any solution satisfying those will now be reported as **near optimal** and the solver will be terminated (note that since this criterion depends on timing, the optimizer will not be run to run deterministic).

All the $\delta$ tolerances discussed above can be adjusted using suitable parameters — see Table 12.1.

Table 12.1: Tolerances for the mixed-integer optimizer.

| Tolerance | Parameter name |
|---|---|
| $\delta_1$ | *MSK_DPAR_MIO_TOL_ABS_RELAX_INT* |
| $\delta_2$ | *MSK_DPAR_MIO_TOL_ABS_GAP* |
| $\delta_3$ | *MSK_DPAR_MIO_TOL_REL_GAP* |
| $\delta_4$ | *MSK_DPAR_MIO_NEAR_TOL_ABS_GAP* |
| $\delta_5$ | *MSK_DPAR_MIO_NEAR_TOL_REL_GAP* |

In Table 12.2 some other common parameters affecting the integer optimizer termination criterion are shown. Please note that if the effect of a parameter is delayed, the associated termination criterion is applied only after some time, specified by the *MSK_DPAR_MIO_DISABLE_TERM_TIME* parameter.

Table 12.2: Other parameters affecting the integer optimizer termination criterion.

| Parameter name | Delayed | Explanation |
|---|---|---|
| *MSK_IPAR_MIO_MAX_NUM_BRANCHES* | Yes | Maximum number of branches allowed. |
| *MSK_IPAR_MIO_MAX_NUM_RELAXS* | Yes | Maximum number of relaxations allowed. |
| *MSK_IPAR_MIO_MAX_NUM_SOLUTIONS* | Yes | Maximum number of feasible integer solutions allowed. |

## 12.4 Speeding Up the Solution Process

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

- Relax the termination criterion: In case the run time is not acceptable, the first thing to do is to relax the termination criterion — see Sec. 12.3 for details.
- Specify a good initial solution: In many cases a good feasible solution is either known or easily computed using problem-specific knowledge. If a good feasible solution is known, it is usually worthwhile to use this as a starting point for the integer optimizer.
- Improve the formulation: A mixed-integer optimization problem may be impossible to solve in one form and quite easy in another form. However, it is beyond the scope of this manual to discuss good formulations for mixed-integer problems. For discussions on this topic see for example *[Wol98]*.

## 12.5 Understanding Solution Quality

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

- The problem status and solution status returned by **MOSEK**, as well as constraint violations in case of suboptimal solutions.
- The *optimality gap* defined as

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

  which measures how much the located solution can deviate from the optimal solution to the problem. The optimality gap can be retrieved through the information item *"MSK_DINF_MIO_OBJ_ABS_GAP"*. Often it is more meaningful to look at the relative optimality gap normalized against the magnitude of the solution.

  $$\epsilon_{\text{rel}} = \frac{|\bar{z} - \underline{z}|}{\max(10^{-10}, |\bar{z}|)}.$$

  The relative optimality gap is available in *"MSK_DINF_MIO_OBJ_REL_GAP"*.

## 12.6 The Optimizer Log

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

```
Presolved problem: 6573 variables, 35728 constraints, 101258 non-zeros
Presolved problem: 0 general integer, 4294 binary, 2279 continuous
Clique table size: 1636
BRANCHES RELAXS   ACT_NDS  DEPTH    BEST_INT_OBJ          BEST_RELAX_OBJ        REL_GAP(%)  TIME
0        1        0        0        NA                    1.8218819866e+07      NA          1.6
0        1        0        0        1.8331557950e+07      1.8218819866e+07      0.61        3.5
0        1        0        0        1.8300507546e+07      1.8218819866e+07      0.45        4.3
Cut generation started.
0        2        0        0        1.8300507546e+07      1.8218819866e+07      0.45        5.3
Cut generation terminated. Time = 1.43
0        3        0        0        1.8286893047e+07      1.8231580587e+07      0.30        7.5
15       18       1        0        1.8286893047e+07      1.8231580587e+07      0.30        10.5
31       34       1        0        1.8286893047e+07      1.8231580587e+07      0.30        11.1
51       54       1        0        1.8286893047e+07      1.8231580587e+07      0.30        11.6
91       94       1        0        1.8286893047e+07      1.8231580587e+07      0.30        12.4
171      174      1        0        1.8286893047e+07      1.8231580587e+07      0.30        14.3
331      334      1        0        1.8286893047e+07      1.8231580587e+07      0.30        17.9

[ ... ]

Objective of best integer solution : 1.825846762609e+07
Best objective bound               : 1.823311032986e+07
Construct solution objective       : Not employed
Construct solution # roundings     : 0
User objective cut value           : 0
Number of cuts generated           : 117
  Number of Gomory cuts            : 108
  Number of CMIR cuts              : 9
Number of branches                 : 4425
Number of relaxations solved       : 4410
Number of interior point iterations: 25
Number of simplex iterations       : 221131
```

The first lines contain a summary of the problem as seen by the optimizer. This is followed by the iteration log. The columns have the following meaning:

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

Following that a summary of the optimization process is printed.

CHAPTER

# THIRTEEN

# PROBLEM ANALYZER

The problem analyzer prints a detailed survey of the

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

of the model.

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

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

```
Analyzing the problem

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

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

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

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

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

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

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

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

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

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

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

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

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

### General Characteristics

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

#### Constraints

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

#### Bounds

- `upper bd` The number of upper bounded variables, $x_j \leq u_j^x$
- `lower bd` The number of lower bounded variables, $l_k^x \leq x_j$
- `ranged` The number of ranged variables, $l_k^x \leq x_j \leq u_j^x$
- `fixed` The number of fixed variables, $l_k^x = x_j = u_j^x$
- `free` The number of free variables

`Variables`

- `cont` The number of continuous variables, $x_j \in \mathbb{R}$
- `bin` The number of binary variables, $x_j \in \{0, 1\}$
- `int` The number of general integer variables, $x_j \in \mathbb{Z}$

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

### Objective

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

The range is displayed using three terms:

- `min |c|` The minimum absolute value among all coeffecients
- `min |c|>0` The minimum absolute value among the nonzero coefficients
- `max |c|` The maximum absolute value among the coefficients

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

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

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

### Linear Constraints

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

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

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

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

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

### Constraint and Variable Bounds

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

### Quadratic Constraints

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

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

### Conic Constraints

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

CHAPTER

# FOURTEEN

# ANALYZING INFEASIBLE PROBLEMS

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

In this section we will

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

## 14.1 Example: Primal Infeasibility

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

As an example of a primal infeasible problem consider the problem of minimizing the cost of transportation between a number of production plants and stores: Each plant produces a fixed number of goods, and each store has a fixed demand that must be met. Supply, demand and cost of transportation per unit are given in Fig. 14.1.

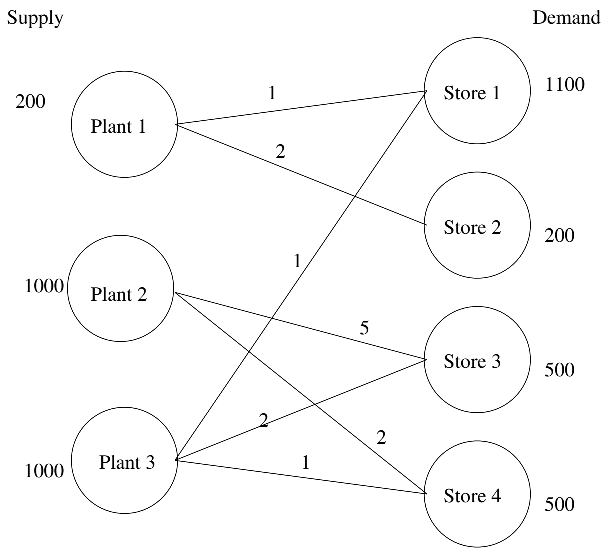


Fig. 14.1: Supply, demand and cost of transportation.

The problem represented in Fig. 14.1 is infeasible, since the total demand

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

exceeds the total supply

$$2200 = 200 + 1000 + 1000$$

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

$$\begin{array}{lccccccccccccccl}
\text{minimize} & x_{11} & + & 2x_{12} & + & 5x_{23} & + & 2x_{24} & + & x_{31} & + & 2x_{33} & + & x_{34} & & \\
\text{subject to} & x_{11} & + & x_{12} & & & & & & & & & & & \leq & 200, \\
 & & & & & x_{23} & + & x_{24} & & & & & & & \leq & 1000, \\
 & & & & & & & & & x_{31} & + & x_{33} & + & x_{34} & \leq & 1000, \\
 & x_{11} & & & & & & & + & x_{31} & & & & & = & 1100, \\
 & & & x_{12} & & & & & & & & & & & = & 200, \\
 & & & & & x_{23} & + & & & & & x_{33} & & & = & 500, \\
 & & & & & & & x_{24} & + & & & & & x_{34} & = & 500, \\
 & & x_{ij} \geq 0. & & & & & & & & & & & & &
\end{array} \tag{14.1}$$

Solving problem (14.1) using **MOSEK** will result in a solution, a solution status and a problem status. Among the log output from the execution of **MOSEK** on the above problem are the lines:

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

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

## 14.2 Locating the cause of Primal Infeasibility

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

- Remove the objective function. This does not change the infeasibility status but simplifies the problem, eliminating any possibility of issues related to the objective function.
- Consider whether your problem has some necessary conditions for feasibility and examine if these are satisfied, e.g. total supply should be greater than or equal to total demand.
- Verify that coefficients and bounds are reasonably sized in your problem.

If the problem is still primal infeasible, some of the constraints must be relaxed or removed completely. The **MOSEK** infeasibility report (Sec. 14.4) may assist you in finding the constraints causing the infeasibility.

Possible ways of relaxing your problem nclude:

- Increasing (decreasing) upper (lower) bounds on variables and constraints.
- Removing suspected constraints from the problem.

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

$$x_{12} = 200$$

makes the problem feasible.

## 14.3 Locating the Cause of Dual Infeasibility

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

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

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

- Adding upper or lower bounds on variables or constraints.
- Removing variables.
- Changing the objective.

### 14.3.1 A cautionary note

The problem

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

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

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

## 14.4 The Infeasibility Report

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

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

### 14.4.1 Example: Primal Infeasibility

We will keep working with the problem (14.1) written in LP format:

Listing 14.1: The code for problem (14.1).

```
\
\ An example of an infeasible linear problem.
\
minimize
 obj: + 1 x11 + 2 x12
      + 5 x23 + 2 x24
      + 1 x31 + 2 x33 + 1 x34
st
  s0: + x11 + x12       <= 200
  s1: + x23 + x24       <= 1000
  s2: + x31 + x33 + x34 <= 1000
```

```
  d1: + x11 + x31        = 1100
  d2: + x12              = 200
  d3: + x23 + x33        = 500
  d4: + x24 + x34        = 500
bounds
end
```

### 14.4.2 Example: Dual Infeasibility

The following problem is dual to (14.1) and therefore it is dual infeasible.

Listing 14.2: The dual of problem (14.1).

```
maximize + 200 y1 + 1000 y2 + 1000 y3 + 1100 y4 + 200 y5 + 500 y6 + 500 y7
subject to
   x11: y1+y4 < 1
   x12: y1+y5 < 2
   x23: y2+y6 < 5
   x24: y2+y7 < 2
   x31: y3+y4 < 1
   x33: y3+y6 < 2
   x34: y3+y7 < 1
bounds
   -inf <= y1 < 0
   -inf <= y2 < 0
   -inf <= y3 < 0
   y4 free
   y5 free
   y6 free
   y7 free
end
```

This can be verified by proving that

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

is a certificate of dual infeasibility (see Sec. 16.1.2) as we can see from this report:

```
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
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
```

Let $y^*$ denote the reported primal solution. **MOSEK** states

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

Since the original objective was maximization, we have that $c^T y^* > 0$. See Sec. 16.1.2 for how to interpret the parameter values in the infeasibility report for a linear program. We see that the variables `y1`, `y3`, `y4`, `y5` and the constraints `x33` and `x34` contribute to infeasibility with non-zero values in the `Activity` column.

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

- Add a lower bound on `y3`. This will directly invalidate the certificate of dual infeasibility.
- Increase the object coefficient of `y3`. Changing the coefficients sufficiently will invalidate the inequality $c^T y^* > 0$ and thus the certificate.
- Add lower bounds on `x11` or `x31`. This will directly invalidate the certificate of infeasibility.

Please note that modifying the problem to invalidate the reported certificate does *not* imply that the problem becomes dual feasible — the 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.

## 14.5 Theory Concerning Infeasible Problems

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

$$\begin{array}{lccccc} \text{minimize} & & & c^T x + c^f & & \\ \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x \end{array} \tag{14.2}$$

where the corresponding dual problem is

$$\begin{array}{lccc} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c & & \\ & +(l^x)^T s_l^x - (u^x)^T s_u^x + c^f & & \\ \text{subject to} & A^T y + s_l^x - s_u^x & = & c, \\ & -y + s_l^c - s_u^c & = & 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \leq 0. & & \end{array} \tag{14.3}$$

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

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

## 14.6 The Certificate of Primal Infeasibility

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

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

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

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

and

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

The well-known *Farkas Lemma* tells us that (14.2) is infeasible if and only if a certificate of primal infeasibility exists.

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

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

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

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

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

## 14.7 The certificate of dual infeasibility

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

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

with negative objective value, where we use the definitions

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

and

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

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

$$\begin{array}{ccccc} & & c^T x^* & < & 0, \\ \bar{l}^c & \leq & Ax^* & \leq & \bar{u}^c, \\ \bar{l}^x & \leq & x^* & \leq & \bar{u}^x \end{array} \tag{14.4}$$

The well-known Farkas Lemma tells us that (14.3) is infeasible if and only if a certificate of dual infeasibility exists.

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

$$x_j^* \leq 0,$$

variable $j$ is involved in the dual infeasibility.

The code in Listing 14.3 will form the repaired problem and solve it.

Listing 14.3: Feasibility repair example.

```
function feasrepairex1(inputfile)

cmd = sprintf('read(%s)', inputfile);
[r,res]=mosekopt(cmd);

res.prob.primalrepair = [];
res.prob.primalrepair.wux = [1,1];
res.prob.primalrepair.wlx = [1,1];
res.prob.primalrepair.wuc = [1,1,1,1];
res.prob.primalrepair.wlc = [1,1,1,1];

param.MSK_IPAR_LOG_FEAS_REPAIR = 3;
[r,res]=mosekopt('minimize primalrepair',res.prob,param);
fprintf('Return code: %d\n',r);

end
```

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 printed out. If the fields `wlx`, `wux`, `wlc` or `wuc` are not specified, they are all assumed to be 1-vectors of appropriate dimensions.

The output from running the commands above is:

```
MOSEK Version 8.0.0.32(BETA) (Build date: 2016-7-17 10:54:55)
Copyright (c) MOSEK ApS, Denmark. WWW: mosek.com
Platform: Linux/64-X86

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

MOSEK Version 8.0.0.32(BETA) (Build date: 2016-7-17 10:54:55)
Copyright (c) MOSEK ApS, Denmark. WWW: mosek.com
Platform: Linux/64-X86

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.
Interior-point 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
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
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.
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.

Optimizer terminated. Time: 0.03
Basic solution summary
  Problem status  : PRIMAL_AND_DUAL_FEASIBLE
  Solution status : OPTIMAL
  Primal.  obj: 4.2500000000e+01    nrm: 6e+02    Viol.  con: 1e-13    var: 0e+00
  Dual.    obj: 4.2499999999e+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 -2.25e+01 on constraint 'c4' (3) with 1.35e+02.
Decreasing the lower bound 6.50e+02 on variable 'x2' (4) with 2.00e+01.
Primal feasibility repair terminated.
Optimizer started.
Presolve started.
Presolve terminated. Time: 0.00
Optimizer terminated. Time: 0.00

Interior-point solution summary
  Problem status  : PRIMAL_AND_DUAL_FEASIBLE
  Solution status : OPTIMAL
  Primal.  obj: -5.6700000000e+03   nrm: 6e+02    Viol.  con: 0e+00    var: 0e+00
  Dual.    obj: -5.6700000000e+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: -5.6700000000e+03   nrm: 6e+02    Viol.  con: 0e+00    var: 0e+00
  Dual.    obj: -5.6700000000e+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
```

reports the optimal repair. In this case it is to increase the upper bound on constraint `c4` by 1.35e2 and decrease the lower bound on variable `x2` by 20.

CHAPTER

# FIFTEEN

# 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 *[Chv83]* 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.

## 15.1 Sensitivity Analysis for Linear Problems

### 15.1.1 The Optimal Objective Value Function

Assume that we are given the problem

$$\begin{array}{rcllll} z(l^c, u^c, l^x, u^x, c) & = & \text{minimize} & & c^T x & \\ & & \text{subject to} & l^c \leq & Ax & \leq u^c, \\ & & & l^x \leq & x & \leq u^x, \end{array} \tag{15.1}$$

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

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

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

$$f_{l_i^c}(\beta) \tag{15.2}$$

shows the optimal objective value as a function of $\beta$. Please note that a change in $\beta$ corresponds to a perturbation in $l_i^c$ and hence (15.2) 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 (15.2) is a piecewise linear and convex function, i.e. its graph may look like in Fig. 15.1 and Fig. 15.2.

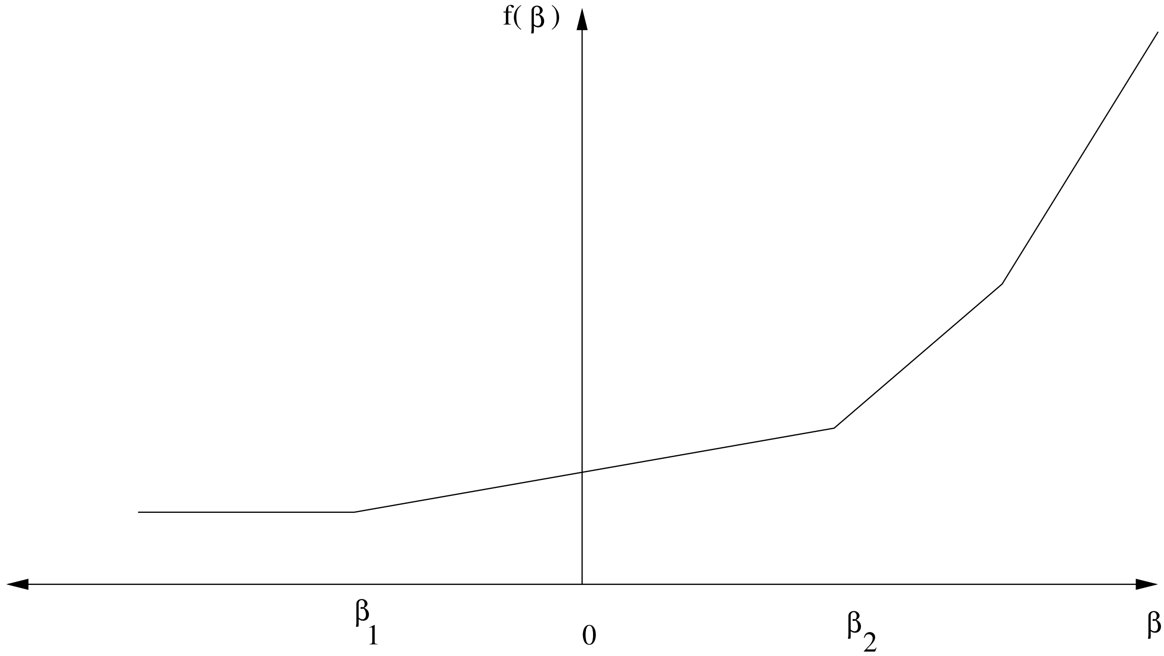


Fig. 15.1: $\beta = 0$ is in the interior of linearity interval.

Clearly, if the function $f_{l_i^c}(\beta)$ does not change much when $\beta$ 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 $\beta$ — 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 $\beta$ around zero. Moreover, we are interested in the *linearity interval*

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

for which

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

Since $f_{l_i^c}$ is not a smooth function $f'_{l_i^c}$ may not be defined at 0, as illustrated in Fig. 15.2. 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 (15.1) as well:

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

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

### 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

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

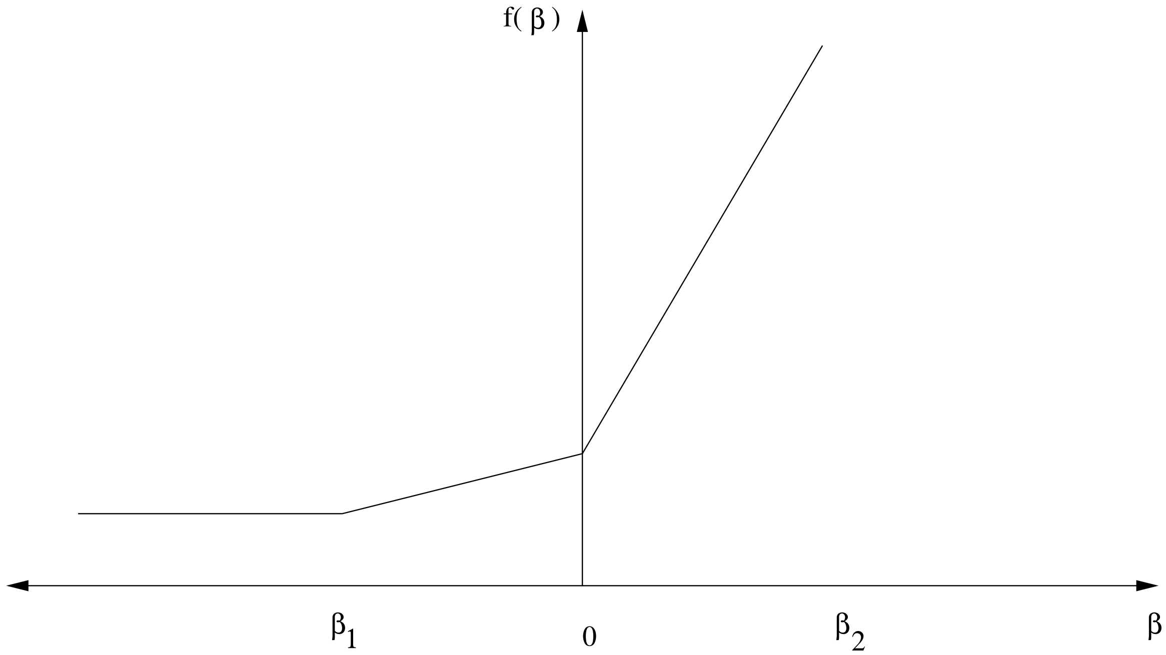


Fig. 15.2: $\beta = 0$ is a breakpoint.

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.

### 15.1.2 The Basis Type Sensitivity Analysis

The classical sensitivity analysis discussed in most textbooks about linear optimization, e.g. *[Chv83]*, is based on an optimal basic solution or, equivalently, on an optimal basis. This method may produce misleading results *[RTV97]* but is **computationally cheap**. Therefore, and for historical reasons, this method is available in **MOSEK**.

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

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

### 15.1.3 The Optimal Partition Type Sensitivity Analysis

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

Given the optimal primal and dual solutions to (15.1), i.e. $x^*$ and $((s_l^c)^*, (s_u^c)^*, (s_l^x)^*, (s_u^x)^*)$ the optimal objective value is given by

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

The left and right shadow prices $\sigma_1$ and $\sigma_2$ for $l_i^c$ are given by this pair of optimization problems:

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

and

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

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

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

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

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

and

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

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

The left and right shadow prices for $c_j$ denoted $\sigma_1$ and $\sigma_2$ respectively are computed as follows:

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

and

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

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

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

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

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

and

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

### 15.1.4 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. 15.3.

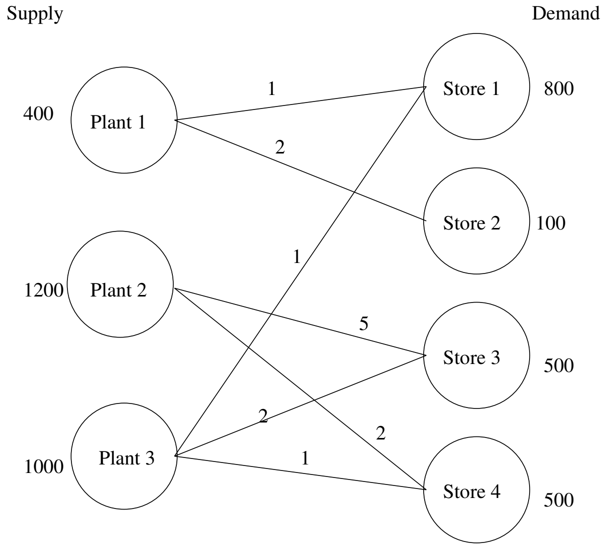


Fig. 15.3: Supply, demand and cost of transportation.

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

$$\begin{array}{ccccccccccccccc} x_{11} & + & x_{12} & & & & & & & & & & & \leq & 400, \\ & & & & x_{23} & + & x_{24} & & & & & & & \leq & 1200, \\ & & & & & & & & x_{31} & + & x_{33} & + & x_{34} & \leq & 1000, \\ x_{11} & & & & & & & + & x_{31} & & & & & = & 800, \\ & & x_{12} & & & & & & & & & & & = & 100, \\ & & & & x_{23} & + & & & & & x_{33} & & & = & 500, \\ & & & & & & x_{24} & + & & & & & x_{34} & = & 500, \\ x_{11}, & & x_{12}, & & x_{23}, & & x_{24}, & & x_{31}, & & x_{33}, & & x_{34} & \geq & 0. \end{array} \tag{15.3}$$

The sensitivity parameters are shown in Table 15.1 and Table 15.2 for the basis type analysis and in Table 15.3 and Table 15.4 for the optimal partition type analysis.

Table 15.1: Ranges and shadow prices related to bounds on constraints and variables: results for the basis type sensitivity analysis.

| Con. | $\beta_1$ | $\beta_2$ | $\sigma_1$ | $\sigma_2$ |
|---|---|---|---|---|
| 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. | $\beta_1$ | $\beta_2$ | $\sigma_1$ | $\sigma_2$ |
| $x_{11}$ | $-\infty$ | 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 15.2: Ranges and shadow prices related to bounds on constraints and variables: results for the optimal partition type sensitivity analysis.

| Con. | $\beta_1$ | $\beta_2$ | $\sigma_1$ | $\sigma_2$ |
|---|---|---|---|---|
| 1 | $-300.00$ | 500.00 | 3.00 | 1.00 |
| 2 | $-700.00$ | $+\infty$ | $-0.00$ | $-0.00$ |
| 3 | $-500.00$ | 500.00 | 3.00 | 1.00 |
| 4 | $-500.00$ | 500.00 | 2.00 | 4.00 |
| 5 | $-100.00$ | 300.00 | 3.00 | 5.00 |
| 6 | $-500.00$ | 700.00 | 3.00 | 5.00 |
| 7 | $-500.00$ | 700.00 | 2.00 | 2.00 |
| Var. | $\beta_1$ | $\beta_2$ | $\sigma_1$ | $\sigma_2$ |
| $x_{11}$ | $-\infty$ | 300.00 | 0.00 | 0.00 |
| $x_{12}$ | $-\infty$ | 100.00 | 0.00 | 0.00 |
| $x_{23}$ | $-\infty$ | 500.00 | 0.00 | 2.00 |
| $x_{24}$ | $-\infty$ | 500.00 | 0.00 | 0.00 |
| $x_{31}$ | $-\infty$ | 500.00 | 0.00 | 0.00 |
| $x_{33}$ | $-\infty$ | 500.00 | 0.00 | 0.00 |
| $x_{34}$ | $-\infty$ | 500.00 | 0.00 | 2.00 |

Table 15.3: Ranges and shadow prices related to the objective coefficients: results for the basis type sensitivity analysis.

| Var. | $\beta_1$ | $\beta_2$ | $\sigma_1$ | $\sigma_2$ |
|---|---|---|---|---|
| $c_1$ | $-\infty$ | 3.00 | 300.00 | 300.00 |
| $c_2$ | $-\infty$ | $\infty$ | 100.00 | 100.00 |
| $c_3$ | $-2.00$ | $\infty$ | 0.00 | 0.00 |
| $c_4$ | $-\infty$ | 2.00 | 500.00 | 500.00 |
| $c_5$ | $-3.00$ | $\infty$ | 500.00 | 500.00 |
| $c_6$ | $-\infty$ | 2.00 | 500.00 | 500.00 |
| $c_7$ | $-2.00$ | $\infty$ | 0.00 | 0.00 |

Table 15.4: Ranges and shadow prices related to the objective coefficients: results for the optimal partition type sensitivity analysis.

| Var. | $\beta_1$ | $\beta_2$ | $\sigma_1$ | $\sigma_2$ |
|---|---|---|---|---|
| $c_1$ | $-\infty$ | 3.00 | 300.00 | 300.00 |
| $c_2$ | $-\infty$ | $\infty$ | 100.00 | 100.00 |
| $c_3$ | $-2.00$ | $\infty$ | 0.00 | 0.00 |
| $c_4$ | $-\infty$ | 2.00 | 500.00 | 500.00 |
| $c_5$ | $-3.00$ | $\infty$ | 500.00 | 500.00 |
| $c_6$ | $-\infty$ | 2.00 | 500.00 | 500.00 |
| $c_7$ | $-2.00$ | $\infty$ | 0.00 | 0.00 |

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

$$\beta \in [0, \beta_1] = [0, 500]$$

then the optimal objective value will decrease by the value

$$\sigma_2 \beta = 1\beta.$$

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

$$\beta \in [0, 300]$$

then the optimal objective value will increase by the value

$$\sigma_1 \beta = 3\beta.$$

## 15.2 Sensitivity Analysis with MOSEK

The following describe sensitivity analysis from the MATLAB toolbox.

### 15.2.1 On bounds

The index of bounds/variables to analyzed for sensitivity are specified in the following subfields of the MATLAB structure `prob`:

- `.prisen.cons.subu` Indexes of constraints, where upper bounds are analyzed for sensitivity.
- `.prisen.cons.subl` Indexes of constraints, where lower bounds are analyzed for sensitivity.
- `.prisen.vars.subu` Indexes of variables, where upper bounds are analyzed for sensitivity.
- `.prisen.vars.subl` Indexes of variables, where lower bounds are analyzed for sensitivity.
- `.duasen.sub` Index of variables where coefficients are analyzed for sensitivity.

For an equality constraint, the index can be specified in either `subu` or `subl`. After calling

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

the results are returned in the subfields `prisen` and `duasen` of `res`.

### 15.2.2 *prisen*

The field `prisen` is structured as follows:

- `.cons`: a MATLAB structure with subfields:
  - `.lr_bl` Left value $\beta_1$ in the linearity interval for a lower bound.
  - `.rr_bl` Right value $\beta_2$ in the linearity interval for a lower bound.
  - `.ls_bl` Left shadow price $s_l$ for a lower bound.
  - `.rs_bl` Right shadow price $s_r$ for a lower bound.
  - `.lr_bu` Left value $\beta_1$ in the linearity interval for an upper bound.
  - `.rr_bu` Right value $\beta_2$ in the linearity interval for an upper bound.
  - `.ls_bu` Left shadow price $s_l$ for an upper bound.
  - `.rs_bu` Right shadow price $s_r$ for an upper bound.
- `.var`: MATLAB structure with subfields:
  - `.lr_bl` Left value $\beta_1$ in the linearity interval for a lower bound on a varable.
  - `.rr_bl` Right value $\beta_2$ in the linearity interval for a lower bound on a varable.
  - `.ls_bl` Left shadow price $s_l$ for a lower bound on a varable.
  - `.rs_bl` Right shadow price $s_r$ for lower bound on a varable.
  - `.lr_bu` Left value $\beta_1$ in the linearity interval for an upper bound on a varable.
  - `.rr_bu` Right value $\beta_2$ in the linearity interval for an upper bound on a varable.
  - `.ls_bu` Left shadow price $s_l$ for an upper bound on a varables.
  - `.rs_bu` Right shadow price $s_r$ for an upper bound on a varables.

#### duasen

The field `duasen` is structured as follows:

- `.lr_c` Left value $\beta_1$ of linearity interval for an objective coefficient.
- `.rr_c` Right value $\beta_2$ of linearity interval for an objective coefficient.
- `.ls_c` Left shadow price $s_l$ for an objective coefficients .

- `.rs_c` Right shadow price $s_r$ for an objective coefficients.

### 15.2.3 Selecting Analysis Type

The type (basis or optimal partition) of analysis to be performed can be selected by setting the parameter *MSK_IPAR_SENSITIVITY_TYPE* to *"MSK_SENSITIVITY_TYPE_BASIS"* or *"MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION"*. as seen in the following example.

#### Example

Consider the problem defined in (15.3). Suppose we wish to perform sensitivity analysis on all bounds and coefficients. The following example demonstrates this as well as the method for changing between basic and full sensitivity analysis.

Listing 15.1: A script to perform sensitivity analysis on problem (15.3).

```
function sensitivity()

clear prob;

% Obtain all symbolic constants
% defined by MOSEK.
[r,res]  = mosekopt('symbcon');
sc       = res.symbcon;

prob.blc = [-Inf, -Inf, -Inf, 800,100,500,500];
prob.buc = [ 400, 1200, 1000, 800,100,500,500];
prob.c   = [1.0,2.0,5.0,2.0,1.0,2.0,1.0]';
prob.blx = [0.0,0.0,0.0,0.0,0.0,0.0,0.0];
prob.bux = [Inf,Inf,Inf,Inf, Inf,Inf,Inf];

subi     = [  1,  1,  2,  2,  3,  3,  3,  4,  4,  5,  6,  6,  7,  7];
subj     = [  1,  2,  3,  4,  5,  6,  7,  1,  5,  6,  3,  6,  4,  7];
val      = [1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0];

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

% analyse upper bound 1:7
prob.prisen.cons.subl = [];
prob.prisen.cons.subu = [1:7];
% analyse lower bound on variables 1:7
prob.prisen.vars.subl = [1:7];
prob.prisen.vars.subu = [];
% analyse coeficient 1:7
prob.duasen.sub = [1:7];
%Select basis sensitivity analysis and optimize.
param.MSK_IPAR_SENSITIVITY_TYPE=sc.MSK_SENSITIVITY_TYPE_BASIS;
[r,res] = mosekopt('minimize echo(0)',prob,param);
results(1) = res;
% Select optimal partition sensitivity analysis and optimize.
param.MSK_IPAR_SENSITIVITY_TYPE=sc.MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION;
[r,res] = mosekopt('minimize echo(0)',prob,param);
results(2) = res;
%Print results
for m = [1:2]
  if m == 1
    fprintf('\nBasis sensitivity results:\n')
  else
    fprintf('\nOptimal partition sensitivity results:\n')
  end
```

```
  fprintf('\nSensitivity for bounds on constraints:\n')
  for i = 1:length(prob.prisen.cons.subl)
    fprintf (...
    'con = %d, beta_1 = %.1f, beta_2 = %.1f, delta_1 = %.1f,delta_2 = %.1f\n', ...
    prob.prisen.cons.subu(i),results(m).prisen.cons.lr_bu(i), ...
    results(m).prisen.cons.rr_bu(i),...
    results(m).prisen.cons.ls_bu(i),...
    results(m).prisen.cons.rs_bu(i));
  end

  for i = 1:length(prob.prisen.cons.subu)
    fprintf (...
    'con = %d, beta_1 = %.1f, beta_2 = %.1f, delta_1 = %.1f,delta_2 = %.1f\n', ...
    prob.prisen.cons.subu(i),results(m).prisen.cons.lr_bu(i), ...
    results(m).prisen.cons.rr_bu(i),...
    results(m).prisen.cons.ls_bu(i),...
    results(m).prisen.cons.rs_bu(i));
  end
  fprintf('Sensitivity for bounds on variables:\n')
  for i = 1:length(prob.prisen.vars.subl)
  fprintf (...
  'var = %d, beta_1 = %.1f, beta_2 = %.1f, delta_1 = %.1f,delta_2 = %.1f\n', ...
   prob.prisen.vars.subl(i),results(m).prisen.vars.lr_bl(i), ...
   results(m).prisen.vars.rr_bl(i),...
   results(m).prisen.vars.ls_bl(i),...
   results(m).prisen.vars.rs_bl(i));
  end

  for i = 1:length(prob.prisen.vars.subu)
    fprintf (...
    'var = %d, beta_1 = %.1f, beta_2 = %.1f, delta_1 = %.1f,delta_2 = %.1f\n', ...
    prob.prisen.vars.subu(i),results(m).prisen.vars.lr_bu(i), ...
    results(m).prisen.vars.rr_bu(i),...
    results(m).prisen.vars.ls_bu(i),...
    results(m).prisen.vars.rs_bu(i));
  end

  fprintf('Sensitivity for coefficients in objective:\n')
  for i = 1:length(prob.duasen.sub)
    fprintf (...
    'var = %d, beta_1 = %.1f, beta_2 = %.1f, delta_1 = %.1f,delta_2 = %.1f\n', ...
    prob.duasen.sub(i),results(m).duasen.lr_c(i), ...
    results(m).duasen.rr_c(i),...
    results(m).duasen.ls_c(i),...
    results(m).duasen.rs_c(i));
  end
end
```

The output from running the example in Listing 15.1 is shown below.

```
Basis sensitivity results:

Sensitivity for bounds on constraints:
con = 1, beta_1 = -300.0, beta_2 = 0.0, delta_1 = 3.0,delta_2 = 3.0
con = 2, beta_1 = -700.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0
con = 3, beta_1 = -500.0, beta_2 = 0.0, delta_1 = 3.0,delta_2 = 3.0
con = 4, beta_1 = -0.0, beta_2 = 500.0, delta_1 = 4.0,delta_2 = 4.0
con = 5, beta_1 = -0.0, beta_2 = 300.0, delta_1 = 5.0,delta_2 = 5.0
con = 6, beta_1 = -0.0, beta_2 = 700.0, delta_1 = 5.0,delta_2 = 5.0
con = 7, beta_1 = -500.0, beta_2 = 700.0, delta_1 = 2.0,delta_2 = 2.0
Sensitivity for bounds on variables:
var = 1, beta_1 = Inf, beta_2 = 300.0, delta_1 = 0.0,delta_2 = 0.0
var = 2, beta_1 = Inf, beta_2 = 100.0, delta_1 = 0.0,delta_2 = 0.0
```

```
var = 3, beta_1 = Inf, beta_2 = 0.0, delta_1 = 0.0,delta_2 = 0.0
var = 4, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 5, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 6, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 7, beta_1 = -0.0, beta_2 = 500.0, delta_1 = 2.0,delta_2 = 2.0
Sensitivity for coefficients in objective:
var = 1, beta_1 = Inf, beta_2 = 3.0, delta_1 = 300.0,delta_2 = 300.0
var = 2, beta_1 = Inf, beta_2 = Inf, delta_1 = 100.0,delta_2 = 100.0
var = 3, beta_1 = -2.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0
var = 4, beta_1 = Inf, beta_2 = 2.0, delta_1 = 500.0,delta_2 = 500.0
var = 5, beta_1 = -3.0, beta_2 = Inf, delta_1 = 500.0,delta_2 = 500.0
var = 6, beta_1 = Inf, beta_2 = 2.0, delta_1 = 500.0,delta_2 = 500.0
var = 7, beta_1 = -2.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0

Optimal partition sensitivity results:

Sensitivity for bounds on constraints:
con = 1, beta_1 = -300.0, beta_2 = 500.0, delta_1 = 3.0,delta_2 = 1.0
con = 2, beta_1 = -700.0, beta_2 = Inf, delta_1 = -0.0,delta_2 = -0.0
con = 3, beta_1 = -500.0, beta_2 = 500.0, delta_1 = 3.0,delta_2 = 1.0
con = 4, beta_1 = -500.0, beta_2 = 500.0, delta_1 = 2.0,delta_2 = 4.0
con = 5, beta_1 = -100.0, beta_2 = 300.0, delta_1 = 3.0,delta_2 = 5.0
con = 6, beta_1 = -500.0, beta_2 = 700.0, delta_1 = 3.0,delta_2 = 5.0
con = 7, beta_1 = -500.0, beta_2 = 700.0, delta_1 = 2.0,delta_2 = 2.0
Sensitivity for bounds on variables:
var = 1, beta_1 = Inf, beta_2 = 300.0, delta_1 = 0.0,delta_2 = 0.0
var = 2, beta_1 = Inf, beta_2 = 100.0, delta_1 = 0.0,delta_2 = 0.0
var = 3, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 2.0
var = 4, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 5, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 6, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 7, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 2.0
Sensitivity for coefficients in objective:
var = 1, beta_1 = Inf, beta_2 = 3.0, delta_1 = 300.0,delta_2 = 300.0
var = 2, beta_1 = Inf, beta_2 = Inf, delta_1 = 100.0,delta_2 = 100.0
var = 3, beta_1 = -2.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0
var = 4, beta_1 = Inf, beta_2 = 2.0, delta_1 = 500.0,delta_2 = 500.0
var = 5, beta_1 = -3.0, beta_2 = Inf, delta_1 = 500.0,delta_2 = 500.0
var = 6, beta_1 = Inf, beta_2 = 2.0, delta_1 = 500.0,delta_2 = 500.0
var = 7, beta_1 = -2.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0
```

CHAPTER

SIXTEEN

# 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.

## 16.1 Linear Optimization

A linear optimization problem can be written as

$$\begin{array}{lccccc} \mbox{minimize} & & & c^T x + c^f & & \\ \mbox{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \end{array} \tag{16.1}$$

where

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

A primal solution $(x)$ is *(primal) feasible* if it satisfies all constraints in (16.1). If (16.1) has at least one primal feasible solution, then (16.1) is said to be (primal) feasible.

In case (16.1) does not have a feasible solution, the problem is said to be *(primal) infeasible*

### 16.1.1 Duality for Linear Optimization

Corresponding to the primal problem (16.1), there is a dual problem

$$\begin{array}{lccc} \mbox{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f & & \\ \mbox{subject to} & A^T y + s_l^x - s_u^x & = & c, \\ & -y + s_l^c - s_u^c & = & 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0. \end{array} \tag{16.2}$$

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

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

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

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

to the dual problem is feasible if it satisfies all the constraints in (16.2). If (16.2) has at least one feasible solution, then (16.2) is *(dual) feasible*, otherwise the problem is *(dual) infeasible*.

### A Primal-dual Feasible Solution

A solution

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

is denoted a *primal-dual feasible solution*, if $(x)$ is a solution to the primal problem (16.1) and $(y, s_l^c, s_u^c, s_l^x, s_u^x)$ is a solution to the corresponding dual problem (16.2).

### The Duality Gap

Let

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

be a primal-dual feasible solution, and let

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

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

$$\begin{array}{l} c^T x^* + c^f - \left\{ (l^c)^T (s_l^c)^* - (u^c)^T (s_u^c)^* + (l^x)^T (s_l^x)^* - (u^x)^T (s_u^x)^* + c^f \right\} \\ = \sum_{i=0}^{m-1} \left[ (s_l^c)_i^* ((x_i^c)^* - l_i^c) + (s_u^c)_i^* (u_i^c - (x_i^c)^*) \right] \\ + \sum_{j=0}^{n-1} \left[ (s_l^x)_j^* (x_j - l_j^x) + (s_u^x)_j^* (u_j^x - x_j^*) \right] \geq 0 \end{array} \tag{16.3}$$

where the first relation can be obtained by transposing and multiplying the dual constraints (16.2) 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.

### An Optimal Solution

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

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

are satisfied.

If (16.1) 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.

### 16.1.2 Infeasibility for Linear Optimization

#### Primal Infeasible Problems

If the problem (16.1) is infeasible (has no feasible solution), **MOSEK** will report a certificate of primal infeasibility: The dual solution reported is the certificate of infeasibility, and the primal solution is undefined.

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

$$\begin{array}{ll} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x \\ \text{subject to} & \\ & A^T y + s_l^x - s_u^x = 0, \\ & -y + s_l^c - s_u^c = 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0, \end{array} \tag{16.4}$$

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 (16.4) 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 (16.4) is unbounded, and that its dual is infeasible. As the constraints to the dual of (16.4) are identical to the constraints of problem (16.1), we thus have that problem (16.1) is also infeasible.

#### Dual Infeasible Problems

If the problem (16.2) is infeasible (has no feasible solution), **MOSEK** will report a certificate of dual infeasibility: The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

$$\begin{array}{lccccc} \text{minimize} & & & c^T x & & \\ \text{subject to} & \hat{l}^c & \leq & Ax & \leq & \hat{u}^c, \\ & \hat{l}^x & \leq & x & \leq & \hat{u}^x, \end{array} \tag{16.5}$$

where

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

and

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

such that

$$c^T x < 0.$$

Such a solution implies that (16.5) is unbounded, and that its dual is infeasible. As the constraints to the dual of (16.5) are identical to the constraints of problem (16.2), we thus have that problem (16.2) is also infeasible.

#### Primal and Dual Infeasible Case

In case that both the primal problem (16.1) and the dual problem (16.2) 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 (16.1) is maximization, i.e.

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

the objective sense of the dual problem changes to minimization, and the domain of all dual variables changes sign in comparison to (16.2). The dual problem thus takes the form

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

This means that the duality gap, defined in (16.3) 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

$$\begin{array}{c} A^T y + s_l^x - s_u^x = 0, \\ -y + s_l^c - s_u^c = 0, \\ s_l^c, s_u^c, s_l^x, s_u^x \leq 0, \end{array} \tag{16.6}$$

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 (16.5) such that $c^T x > 0$.

## 16.2 Conic Quadratic Optimization

*Conic quadratic optimization* is an extension of linear optimization (see Sec. 16.1) allowing conic domains to be specified for subsets of the problem variables. A conic quadratic optimization problem can be written as

$$\begin{array}{lcccccc} \text{minimize} & & & c^T x + c^f & & \\ \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \\ & & & x \in \mathcal{K}, & & \end{array} \tag{16.7}$$

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

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

where $x = (x^1, \ldots, x^p)$ is a partition of the problem variables. Please note that the $n$-dimensional Euclidean space $\mathbb{R}^n$ is a cone itself, so simple linear variables are still allowed.

**MOSEK** supports only a limited number of cones, specifically:

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

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

- The rotated quadratic cone:

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

Although these cones may seem to provide only limited expressive power they can be used to model a wide range of problems as demonstrated in *[MOSEKApS12]*.

### 16.2.1 Duality for Conic Quadratic Optimization

The dual problem corresponding to the conic quadratic optimization problem (16.7) is given by

$$\begin{array}{ll} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f \\ \text{subject to} & \\ & A^T y + s_l^x - s_u^x + s_n^x = c \\ & -y + s_l^c - s_u^c = 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0, \\ & s_n^x \in \mathcal{K}^*, \end{array} \tag{16.8}$$

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

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

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

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

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

- The quadratic cone:

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

- The rotated quadratic cone:

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

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

### 16.2.2 Infeasibility for Conic Quadratic Optimization

In case **MOSEK** finds a problem to be infeasible it reports a certificate of infeasibility. This works exactly as for linear problems (see Sec. 16.1.2).

#### Primal Infeasible Problems

If the problem (16.7) is infeasible, **MOSEK** will report a certificate of primal infeasibility: The dual solution reported is the certificate of infeasibility, and the primal solution is undefined.

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

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

such that the objective value is strictly positive.

### Dual infeasible problems

If the problem (16.8) is infeasible, **MOSEK** will report a certificate of dual infeasibility: The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

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

where

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

and

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

such that the objective value is strictly negative.

## 16.3 Semidefinite Optimization

*Semidefinite optimization* is an extension of conic quadratic optimization (see Sec. 16.2) allowing positive semidefinite matrix variables to be used in addition to the usual scalar variables. A semidefinite optimization problem can be written as

$$\begin{array}{lccccll} \text{minimize} & & & \sum_{j=0}^{n-1} c_j x_j + \sum_{j=0}^{p-1} \left\langle \overline{C}_j, \overline{X}_j \right\rangle + c^f & & & \\ \text{subject to} & l_i^c & \leq & \sum_{j=0}^{n-1} a_{ij} x_j + \sum_{j=0}^{p-1} \left\langle \overline{A}_{ij}, \overline{X}_j \right\rangle & \leq & u_i^c, & i = 0, \ldots, m-1 \\ & l_j^x & \leq & x_j & \leq & u_j^x, & j = 0, \ldots, n-1 \\ & & & x \in \mathcal{K}, \overline{X}_j \in \mathcal{S}_+^{r_j}, & & & j = 0, \ldots, p-1 \end{array} \tag{16.9}$$

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

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

With semidefinite optimization we can model a wide range of problems as demonstrated in *[MOSEKApS12]*.

### 16.3.1 Duality for Semidefinite Optimization

The dual problem corresponding to the semidefinite optimization problem (16.9) is given by

$$\begin{array}{lll} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f & \\ \text{subject to} & & \\ & c - A^T y + s_u^x - s_l^x = s_n^x, & \\ & \overline{C}_j - \sum_{i=0}^{m} y_i \overline{A}_{ij} = \overline{S}_j, & j = 0, \ldots, p-1 \\ & s_l^c - s_u^c = y, & \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0, & \\ & s_n^x \in \mathcal{K}^*, \quad \overline{S}_j \in \mathcal{S}_+^{r_j}, & j = 0, \ldots, p-1 \end{array} \tag{16.10}$$

where $A \in \mathbb{R}^{m\times n}$, $A_{ij} = a_{ij}$, which is similar to the dual problem for conic quadratic optimization (see Sec. 16.2.1), except for the addition of dual constraints

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

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

### 16.3.2 Infeasibility for Semidefinite Optimization

In case **MOSEK** finds a problem to be infeasible it reports a certificate of the infeasibility. This works exactly as for linear problems (see Sec. 16.1.2).

#### Primal Infeasible Problems

If the problem (16.9) is infeasible, **MOSEK** will report a certificate of primal infeasibility: The dual solution reported is a certificate of infeasibility, and the primal solution is undefined.

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

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

such that the objective value is strictly positive.

#### Dual Infeasible Problems

If the problem (16.10) is infeasible, **MOSEK** will report a certificate of dual infeasibility: The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

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

where

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

and

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

such that the objective value is strictly negative.

## 16.4 Quadratic and Quadratically Constrained Optimization

A convex quadratic and quadratically constrained optimization problem has the form

$$\begin{array}{lrcccl} \text{minimize} & & & \frac{1}{2}x^TQ^ox + c^Tx + c^f & & \\ \text{subject to} & l_k^c & \leq & \frac{1}{2}x^TQ^kx + \sum_{j=0}^{n-1} a_{kj}x_j & \leq & u_k^c, \quad k = 0, \ldots, m-1, \\ & l_j^x & \leq & x_j & \leq & u_j^x, \quad j = 0, \ldots, n-1, \end{array} \tag{16.11}$$

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

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

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

### 16.4.1 A Recommendation

Any convex quadratic optimization problem can be reformulated as a conic quadratic optimization problem, see *[MOSEKApS12]* and in particular *[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 modeller can do a better reformulation than the automatic method because the modeller 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.

### 16.4.2 Duality for Quadratic and Quadratically Constrained Optimization

The dual problem corresponding to the quadratic and quadratically constrained optimization problem (16.11) is given by

$$\begin{array}{ll} \text{maximize} & (l^c)^Ts_l^c - (u^c)^Ts_u^c + (l^x)^Ts_l^x - (u^x)^Ts_u^x + \frac{1}{2}x^T\left\{\sum_{k=0}^{m-1} y_kQ^k - Q^o\right\}x + c^f \\ \text{subject to} & A^Ty + s_l^x - s_u^x + \left\{\sum_{k=0}^{m-1} y_kQ^k - Q^o\right\}x = c, \\ & -y + s_l^c - s_u^c = 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0. \end{array} \tag{16.12}$$

The dual problem is related to the dual problem for linear optimization (see Sec. 16.1.1), 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 (16.11) and the dual problem (16.12).

### 16.4.3 Infeasibility for Quadratic and Quadratically Constrained Optimization

In case **MOSEK** finds a problem to be infeasible it reports a certificate of infeasibility. This works exactly as for linear problems (see Sec. 16.1.2).

#### Primal Infeasible Problems

If the problem (16.11) with all $Q^k = 0$ is infeasible, **MOSEK** will report a certificate of primal infeasibility. As the constraints are the same as for a linear problem, the certificate of infeasibility is the same as for linear optimization (see Sec. 16.1.2).

#### Dual Infeasible Problems

If the problem (16.12) with all $Q^k = 0$ is dual infeasible, **MOSEK** will report a certificate of dual infeasibility. The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

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

where

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

and

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

such that the objective value is strictly negative.

## 16.5 General Convex Optimization

The general nonlinear optimizer (which may be available for all or some types of nonlinear problems depending on the interface), solves smooth (twice differentiable) convex nonlinear optimization problems of the form

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

where

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

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

This means that the $i$-th constraint has the form

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

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

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

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

### 16.5.1 Duality for General convex Optimization

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

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

and the dual problem is given by

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

which is equivalent to

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

In this context we use the following definition for scalar functions

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

and accordingly for vector functions

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

CHAPTER

# SEVENTEEN

# TOOLBOX REFERENCE

## 17.1 Command Reference

The **MOSEK** toolbox provides a set of functions to interface to the **MOSEK** solver.

### Main interface

*mosekopt* is the main interface to **MOSEK**.

### Helper functions

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

- *msklpopt* : Solves linear optimization problems.
- *mskqpopt* : Solves quadratic optimization problems.
- *mskenopt* : Solves entropy optimization problems.
- *mskgpopt* : Solves geometric optimization problems.
- *mskscopt* : Solves separable convex optimization problems.

### I/O

- *mskgpwri* : Write a geometric optimization problem to file.
- *mskgpread*: Read a geometric optimization problem from file.

### Options

- *mskoptimget* : Get the solver parameters.
- *mskoptimset* : Set the solver parameters.

### MATLAB optimization toolbox compatible functions.

- *linprog* : Solves linear optimization problems.
- *quadprog* : Solves quadratic optimization problems.
- *intlinprog* : Solves linear optimization problems with integer variables.

- *lsqlin* : Solves least-squares with linear constraints.
- *lsqnonneg* : Solves least-squares with non-negativity constraints.

### 17.1.1 Main Interface

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

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

The following command strings are recognized for the *cmd* parameter:

- `anapro`: Runs the problem analyzer.
- `echo(n)`: Controls how much information is echoed to the screen. `n` must be a nonnegative integer, where 0 means that no information is displayed and 3 means that all information is displayed.
- `info`: Return the complete task information database in the field *info* of a *res* struct.
- `param`: Return the complete parameter database in res.param.
- `primalrepair`: Performs a primal feasibility repair. See Sec. 14 for details.
- `maximize`: Maximize the objective.
- `max` : Sets the objective sense (similar to `.maximize`), without performing an optimization.
- `minimize`: Minimize the objective.
- `min`: Sets the objective sense (similar to `.minimize`), without performing an optimization.
- `nokeepenv`: Delete the **MOSEK** environment after each run. This can increase the license checkout overhead significantly and is therefore only intended as a debug feature.
- `read(name)`: Request that data is read from a file `name`.
- `statuskeys(n)`: Controls the format of status keys (problem status, solution status etc.) in the returned problem:
  - `statuskeys(0)` – all the status keys are returned as strings,
  - `statuskeys(1)` – all the status keys are returned as numeric codes.
- `symbcon`: Return the symbcon data structure in `res.symbcon`. See structure *symbcon* for details.
- `write(name)`: Write problem to the file `name`.
- `log(name)`: Write solver log-output to the file `name`.
- `version`: Return the **MOSEK** version numbers in `res.version`.

**Parameters**

- `cmd [in]` (string) – The commands to be executed. By default it takes the value *minimize*.
- `prob [in]` (prob) – [*optional*] a structure containing the problem data.
- `param [in]` (struct) – [*optional*] a structure which is used to specify algorithmic parameters to **MOSEK**. The fields of `param` must be valid **MOSEK** parameter names. Moreover, the values corresponding to the fields must be of a valid type, i.e. the value of a string parameter must be a string, the value of an integer paramter must be an integer etc.
- `callback [in]` (callback) – [*optional*] A MATLAB structure defining call-back data and functions.

**Return**

- `rcode` (rescode) – Return code. The interpretation of the value of the return code is listed in Sec. 17.6.
- `res` (res) – [*optional*] Solution obtained by the interior-point algorithm.

### 17.1.2 Helper Functions

`res = msklpopt(c, a, blc, buc, blx, bux, param, cmd)`

Solves a linear optimization problem of the form

$$\begin{array}{ll} min & c^T x \\ st. & \\ & blc \leq Ax \leq buc \\ & bux \leq x \leq bux. \end{array}$$

**Note:** *lc=[]* and *buc=[]* means that the lower and upper bounds are plus and minus infinite respectively. The same interpretation is used for `blx` and `bux`. Note `-inf` is allowed in `blc` and `blx`. Similarly, *inf* is allowed in *buc* and *bux*.

**Parameters**

- `c [in]` (double[]) – The objective function vector.
- `a [in]` (double[][]) – A (preferably sparse) matrix.
- `blc [in]` (double[]) – Constraints lower bounds.
- `buc [in]` (double[]) – Constraints upper bounds.
- `blx [in]` (double[]) – Variables lower bounds.
- `bux [in]` (double[]) – Variables upper bounds.
- `param [in]` (list) – New **MOSEK** parameters.
- `cmd [in]` (list) – [*optional*] The command list. See *mosekopt* for a list of available commands.

**Return** `res` (res) – [*optional*] Solution information.

`res = mskqpopt(q, c, a, blc, buc, blx, bux, param, cmd)`

Solves the optimization problem

$$\begin{array}{ll} \min & \frac{1}{2}x^T Qx + c^T x \\ st. & \\ & blc \leq Ax \leq buc \\ & blx \leq x \leq bux \end{array}$$

**Note:** *blc=[]* and *buc=[]* means that the lower and upper bounds are plus and minus infinite respectively. The same interpretation is used for *blx* and *bux*. Note *-inf* is allowed in *blc* and *blx*. Similarly, *inf* is allowed in *buc* and *bux*.

**Parameters**

- `q` (double[]) – It is assumed that q is a *symmetric positive semi-definite matrix*.
- `c [in]` (double[]) – A vector.
- `a` (float[][]) – A (preferably) sparse matrix.
- `blc [in]` (double[]) – Constraints lower bounds.

- `buc [in]` (double[]) – Constraints upper bounds
- `blx [in]` (double[]) – Variables lower bounds
- `bux [in]` (double[]) – Variables upper bounds
- `param [in]` (list) – **MOSEK** parameters.
- `cmd [in]` (string) – [*optional*] The command list. See *mosekopt* for a list of available commands.

**Return** `res` (res) – [*optional*] Solution information.

`res = mskenopt(d, c, a, blc, buc, param, cmd)`

Solves the entropy optimization problem

$$\begin{array}{ll} \min & d^T(\sum x_i ln(x_i)) + c^T x \\ \text{s.t.} & blc \leq Ax \leq buc \\ & x \in \mathbb{R}^n_+ \end{array}$$

It is required that $d \geq 0.0$.

**Parameters**

- `d [in]` (double[]) – A vector of non negative values.
- `c [in]` (double[]) – A vector.
- `a [in]` (double[][]) – A (preferably) sparse matrix.
- `blc [in]` (double[]) – [optional] Constraints lower bounds.
- `buc [in]` (double[]) – [optional] Constraints Upper bounds.
- `param [in]` (list) – [optional] **MOSEK** parameters.
- `cmd [in]` (list) – [optional] **MOSEK** commands. See *mosekopt* for a list of available commands.

**Return** `res` (res) – [*optional*] Solution information.

`res = mskgpopt(c, a, map, param, cmd)`

Solves the posynomial version of the geometric optimization problem in exponential form:

$$\begin{array}{ll} \min & \log(sum(k \in find(map == 0), c(k) * exp(a(k,:) * x)) \\ st. & \log(sum(k \in find(map == i), c(k) * exp(a(k,:) * x)) <= 0, fork = 1, ..., max(map) \end{array} \tag{17.1}$$

It is required that $c > 0.0$. See Sec. 7.3.

**Parameters**

- `c [in]` (double[]) – A vector.
- `a [in]` (double[][]) – A (preferably) sparse matrix.
- `map` (int[][]) – Corresponds to the set $J$ in Sec. 7.3.
- `param [in]` (list) – [optional] **MOSEK** parameters.
- `cmd [in]` (list) – [optional] The command list. See *mosekopt* for a list of available commands.

**Return** `res` (res) – [*optional*] Solution information.

**See also** *mskgpwri*, *mskgpread*

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

Solves separable convex optimization problems on the form

$$\begin{array}{ll} \min & c^T x + \sum_j f_j(x_j) \\ \text{s.t.} & blc \leq a_x + \sum_j g_{kj}(x_j) \leq buc(k), k = 1, ..., size(a) \\ & blx \leq x \leq bux \end{array}$$

The nonlinear functions $f_j$ and $g_{kj}$ are specified using `opr`, `opri`, `oprj` `oprf`, `oprg` as follows. For all `k` between 1 and `length(opri)` then following nonlinear expression

```
if opr(k,:)=='ent'
  oprf(k) * x(oprj(k)) * log(x(oprj(k)))
elseif if opr(k,:)=='exp'
  oprf(k) * exp(oprg(k)*x(oprj(k)))
elseif if opr(k,:)=='log'
  oprf(k) * log(x(oprj(k)))
elseif if opr(k,:)=='pow'
  oprf(k) * x(oprj(k))^oprg(k)
else
  An invalid operator has been specified.
```

Is added to the objective if `opri(k)=0`. Otherwise it is added to constraint `opri(k)`.

**Parameters**

- `c [in]` (double[]) – Is a vector.
- `a [in]` (double[][]) – Is a (preferably) sparse matrix.
- `blc [in]` (double[]) – [optional] Lower bounds on constraints.
- `buc [in]` (double[]) – [optional] Upper bounds on constraints.
- `blx [in]` (double[]) – [optional] Lower bounds on variables.
- `bux [in]` (double[]) – [optional] Upper bounds on variables.
- `param [in]` (list) – [optional] **MOSEK** parameters.
- `cmd [in]` (list) – [optional] The command list. See *mosekopt* for a list of available commands.

**Return** `res` (res) – [*optional*] Solution information.

### 17.1.3 I/O

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

This function reads a Geometric Programming (gp) problem from a file compatible with the *mskenopt* command tool.

**Parameters** `filename [in]` (string) – The name of the file to read.

**Return**

- `c` (double[]) – Objective function coefficients. See problem (17.1).
- `a` (double[]) – Linear constraints coefficients. See problem (17.1).
- `map` (struct) – Data in the same format accepted by *mskgpopt*.

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

This function writes a Geometric Programming (gp) problem to a file in a format compatible with the *mskenopt* command tool.

**Parameters**

- `c [in]` (double[]) – Objective function coefficients. See problem (17.1).
- `a [in]` (double[]) – Linear constraints coefficients. See problem (17.1).
- `map [in]` (struct) – Data in the same format accepted by *mskgpopt*.
- `filename` (string) – The output file name.

### 17.1.4 Options

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

Obtains a value of an optimization parameter. See the *mskoptimset* function for which parameters that can be set.

**Parameters**

- `options [in]` (struct) – The optimization options structure.
- `param [in]` (string) – Name of the optimization parameter for which the value should be obtained.
- `default [in]` (string) – [*optional*] If `param` is not defined, the value of `default` is returned instead.

**Return** `val` (list) – Value of the required option. If the option does not exist, then `[]` is returned unless the value `default` is defined in which case the default value is returned.

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

Obtains and modifies the optimization options structure. Only a subset of the fields in the optimization structure recognized by the MATLAB optimization toolbox is recognized by **MOSEK**. In addition the optimization options structure can be used to modify all the **MOSEK** specific parameters defined in Sec. 17.4.

- `.Diagnostics` Used to control how much diagnostic information is printed. Following values are accepted:

| | |
|---|---|
| `off` | No diagnostic information is printed. |
| `on` | Diagnostic information is printed. |

- `.Display` Defines what information is displayed. The following values are accepted:

| | |
|---|---|
| `off` | No output is displayed. |
| `iter` | Some output is displayed for each iteration. |
| `final` | Only the final output is displayed. |

- `.MaxIter` Maximum number of iterations allowed.
- `.Write` A filename to write the problem to. If equal to the empty string no file is written. E.g the option `Write(myfile.opf)` writes the file `myfile.opf` in the `opf` format.

**Parameters**

- `arg1 [in]` (None) – [*optional*] Is allowed to be any of the following two things [in]:
  - Any string The same as using no argument.
  - A structure The argument is assumed to be a structure containing options, which are copied to the return options.
- `param1 [in]` (string) – [*optional*] A string containing the name of a parameter that should be modified.
- `value1 [in]` (None) – [*optional*] The new value assigned to the parameter with the name `param1`.
- `param2 [in]` (None) – [*optional*] See `param1`.
- `value2 [in]` (None) – [*optional*] See `value1`.

**Return** `options` (struct) – The updated optimization options structure.

### 17.1.5 MATLAB Optimization Toolbox Compatible Functions.

`x, fval, exitflag, output = intlinprog(f, A, b, B, c, x0, options)`
`x, fval, exitflag, output = intlinprog(problem)`

Solves the binary linear optimization problem:

$$\begin{array}{ll} \text{minimize} & f^T x \\ \text{subject to} & Ax \le b, \\ & Bx = c, \\ & x \in \{0,1\}^n \end{array}$$

**Parameters**

- `f [in]` (double[]) – The objective function.
- `A [in]` (double[][]) – Constraint matrix for the inequalities. Use `A=[]` if there are no inequalities.
- `b [in]` (double[]) – Right-hand side for the inequalities. Use `b=[]` if there are no inequalities.
- `B [in]` (double[][]) – [*optional*] Constraint matrix for the equalities. Use `B=[]` if there are no equalities.
- `c [in]` (double[]) – [*optional*] Right-hand side for the equalities. Use `c=[]` if there are no equalities.
- `x0 [in]` (double[]) – [*optional*] A feasible starting point.
- `options [in]` (struct) – [*optional*] An optimization options structure. See the *mskoptimset* function for the definition of the optimization options structure. *intlinprog* uses the options
  - `.Diagnostics`
  - `.Display`
  - `.MaxTime` The maximum number of seconds in the solution-time
  - `.MaxNodes` The maximum number of branch-and-bounds allowed
  - `.Write` Filename of problem file to save.
- `problem [in]` (struct) – A structure containing the fields `f, A, b, B, c, x0` and `options`.

**Return**

- `x` (double[]) – The solution $x$.
- `fval` (double) – The objective $f^T x$.
- `exitflag` (int) – A number which has the interpretation:
  - 1 The function returned an integer feasible solution.
  - $-2$ The problem is infeasible.
  - $-4$ `maxNodes` reached without converging.
  - $-5$ `maxTime` reached without converging.

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

Solves the linear optimization problem:

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

**Parameters**

- f [in] (double[]) – The objective function.
- A [in] (double[][]) – Constraint matrix for the inequalities. Use $A = []$ if there are no inequalities.
- b [in] (double[]) – Right-hand side for the inequalities. Use $b = []$ if there are no inequalities.
- B [in] (double[][]) – [*optional*] Constraint matrix for the equalities. Use $B = []$ if there are no equalities.
- c [in] (double[]) – [*optional*] Right-hand side for the equalities. Use $c = []$ if there are no equalities.
- l [in] (double[]) – [*optional*] Lower bounds on the variables. Use $-\infty$ to represent infinite lower bounds.
- u [in] (double[]) – [*optional*] Upper bounds on the variables. Use $\infty$ to represent infinite upper bounds.
- x0 [in] (double[]) – [*optional*] An initial guess for the starting point. Only used for the primal simplex algorithm. For more advanced warm-starting (e.g., using dual simplex), use *mosekopt* directly.
- options [in] (struct) – [*optional*] An optimization options structure. See the *mskoptimset* function for the definition of the optimization options structure. *linprog* uses the options
  - .Diagnostics
  - .Display
  - .MaxIter
  - .Simplex Valid values are 'on', 'primal' or 'dual'. If Simplex is 'on' then **MOSEK** will use either a primal or dual simplex solver (similar as specifying *"MSK_OPTIMIZER_FREE_SIMPLEX"* in *mosekopt*; otherwise either a primal or dual simplex algorithm is used. Note, that the 'primal' and 'dual' values are specific for the **MOSEK** interface, and not present in the standard MATLAB version.
  - .Write Filename of problem file (e.g., 'prob.opf') to be saved. This is useful for reporting bugs or problems.
- problem [in] (struct) – structure containing the fields f, A, b, B, c, l, u, x0 and options.
- output [in] (struct) – A struct with the following fields
  - .iterations Number of iterations spent to reach the optimum.
  - .algorithm Always defined as 'large-scale [in]: interior-point'.
- lambda [in] (struct) – A struct with the following fields
  - .lower Lagrange multipliers for lower bounds $l$.
  - .upper Lagrange multipliers for upper bounds $u$.
  - .ineqlin Lagrange multipliers for the inequalities.
  - .eqlin Lagrange multipliers for the equalities.

**Return**

- x (double[]) – The optimal $x$ solution.
- fval (double) – The optimal objective value, i.e. $f^T x$.
- exitflag (int) – A number which has the interpretation [in]:

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

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

Solves the linear least squares problem:

$$
\begin{array}{ll}
\text{minimize} & \frac{1}{2}\left\|Cx - d\right\|_2^2 \\
\text{subject to} & Ax \le b, \\
 & Bx = c, \\
 & l \le x \le u.
\end{array}
\tag{17.2}
$$

**Parameters**

- `C [in]` (double[][]) – A matrix. See problem (17.2) for the purpose of the argument.
- `d [in]` (double[]) – A vector. See problem (17.2) for the purpose of the argument.
- `A [in]` (double[][]) – Constraint matrix for the inequalities. Use $A = []$ if there are no inequalities.
- `b [in]` (double[]) – Right-hand side for the inequalities. Use $b = []$ if there are no inequalities.
- `B [in]` (double[][]) – [*optional*] Constraint matrix for the equalities. Use $B = []$ if there are no equalities.
- `c [in]` (double[]) – [*optional*] Right-hand side for the equalities. Use $c = []$ if there are no equalities.
- `l [in]` (double[]) – [*optional*] Lower bounds on the variables. Use $-\infty$ to represent infinite lower bounds.
- `u [in]` (double[]) – [*optional*] Upper bounds on the variables. Use $\infty$ to represent infinite lower bounds.
- `x0 [in]` (double[]) – [*optional*] An initial guess for the starting point. This information is ignored by **MOSEK**
- `options [in]` (struct) – [*optional*] An optimization options structure. See the function *mskoptimset* function for the definition of the optimization options structure. *lsqlin* uses the options
  - `.Diagnostics`
  - `.Display`
  - `.MaxIter`
  - `.Write`

**Return**

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

- `.iterations` Number of iterations spent to reach the optimum.
- `.algorithm` Always defined as 'large-scale: interior-point'.

- `lambda` (struct) –
  - `.lower` Lagrange multipliers for lower bounds $l$.
  - `.upper` Lagrange multipliers for upper bounds $u$.
  - `.ineqlin` Lagrange multipliers for inequalities.
  - `.eqlin` Lagrange multipliers for equalities.

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

Solves the linear least squares problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\left\|Cx - d\right\|_2^2 \\ \text{subject to} & x \geq 0. \end{array} \tag{17.3}$$

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

**Parameters**

- `C [in]` (double[][]) – See problem (17.3).
- `d [in]` (double[]) – See problem (17.3).
- `x0 [in]` (double[]) – [*optional*] An initial guess for the starting point. This information is ignored by **MOSEK**
- `options [in]` (struct) – [*optional*] An optimizations options structure. See the *mskoptimset* function for the definition of the optimization options structure. *lsqlin* uses the options
  - `.Diagnostics`
  - `.Display`
  - `.MaxIter`
  - `.Write`

**Return**

- `x` (double[]) – The $x$ solution.
- `resnorm` (double) – The squared norm of the optimal residuals, i.e. .. math:: left| Cx-d right|^2

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

  - `.ineqlin` Lagrange multipliers for inequalities.
  - `.eqlin` Lagrange multipliers for equalities.

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

Solves the quadratic optimization problem:

$$
\begin{array}{ll}
\text{minimize} & \frac{1}{2}x^T H x + f^T x \\
\text{subject to} & Ax \leq b, \\
 & Bx = c, \\
 & l \leq x \leq u.
\end{array} \tag{17.4}
$$

**Parameters**

- `H [in]` (double[][]) – Hessian of the objective function. $H$ must be a symmetric matrix. Contrary to the MATLAB optimization toolbox, **MOSEK** handles only the cases where $H$ is positive semidefinite. On the other hand **MOSEK** always computes a global optimum, i.e. the objective function has to be strictly convex.
- `f [in]` (double[]) – See (17.4) for the definition.
- `A [in]` (double[][]) – Constraint matrix for the inequalities. Use $A = []$ if there are no inequalities.
- `b [in]` (double[]) – Right-hand side for the nequalities. Use $b = []$ if there are no inequalities.
- `B [in]` (double[][]) – [*optional*] Constraint matrix for the equalities. Use $B = []$ if there are no equalities.
- `c [in]` (double[]) – [*optional*] Right-hand side for the equalities. Use $c = []$ if there are no equalities.
- `l [in]` (double[]) – [*optional*] Lower bounds on the variables. Use $-\infty$ to represent infinite lower bounds.
- `u [in]` (double[]) – [*optional*] Upper bounds on the variables. Use $\infty$ to represent infinite upper bounds.
- `x0 [in]` (double[]) – [*optional*] An initial guess for the starting point. This information is ignored by **MOSEK**
- `options [in]` (struct) – [*optional*] An optimization options structure. See the *`mskoptimset`* function for the definition of the optimizations options structure. *`quadprog`* uses the options
  - `.Diagnostics`
  - `.Display`
  - `.MaxIter`
  - `.Write`

**Return**

- `x` (double[]) – The $x$ solution.
- `fval` (double) – The optimal objective value i.e. $\frac{1}{2}x^T H x + f^T x$.
- `exitflag` (int) – A scalar which has the interpretation:
  - $< 0$ The problem is likely to be either primal or dual infeasible.
  - $= 0$ The maximum number of iterations was reached.
  - $> 0$ $x$ is an optimal solution.
- `output` (struct) – A structure with the following fields
  - `.iterations` Number of iterations spent to reach the optimum.

  - `.algorithm` Always defined as 'large-scale: interior-point'.
- `lambda` (struct) – A structure with the following fields
  - `.lower` Lagrange multipliers for lower bounds $l$.
  - `.upper` Lagrange multipliers for upper bounds $u$.
  - `.ineqlin` Lagrange multipliers for inequalities.
  - `.eqlin` Lagrange multipliers for equalities.

## 17.2 Data Structures and Notation

The data structures employed by **MOSEK** are discussed in this section, along with the used *notation*.

The data structures and types used are the following:

- *prob*
- *names*
- *cones*
- *barc*
- *bara*
- *solver_solutions*
- *solution*
- *res*
- *prisen*
- *cprisen*
- *uprisen*
- *duasen*
- *info*
- *symbcon*
- *callback*

### 17.2.1 Notation

**MOSEK** solves linear, quadratic, quadratically constrained, and conic optimization problems. The simplest of those is a linear problem, which is posed in **MOSEK** as

$$\begin{array}{lccccl}
\text{minimize} & & & \sum_{j=1}^{n} c_j x_j + c^f & & \\
\text{subject to} & l_i^c & \leq & \sum_{j=1}^{n} a_{ij} x_j & \leq & u_i^c, \quad i = 1, \ldots, m, \\
 & l_j^x & \leq & x_j & \leq & u_j^x, \quad j = 1, \ldots, n.
\end{array}$$

An extension is a linear conic problem where the variables can belong to quadratic or semidefinite cones. A conic problem in **MOSEK** has the form

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

where the conic constraint

$$x \in \mathcal{K} \tag{17.5}$$

means that a partitioning of $x$ belongs to a set of quadratic cones (elaborated below). Further, the problem has $p$ symmetric positive semidefinite variables $\overline{X}_j \in \mathcal{S}_+^{r_j}$ of dimension $r_j$ with symmetric coefficient matrices $\overline{C}_j \in \mathcal{S}^{r_j}$ and $\overline{A}_{i,j} \in \mathcal{S}^{r_j}$.

Alternatively, **MOSEK** can solve convex quadratically constrained quadratic problems

$$\begin{array}{lccccl}
\text{minimize} & & & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}^o x_i x_j + \sum_{j=1}^{n} c_j x_j + c^f & & \\
\text{subject to} & l_i^c & \leq & \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} q_{jk}^i x_j x_k + \sum_{j=1}^{n} a_{ij} x_j & \leq & u_i^c, \quad i = 1, \ldots, m, \\
 & l_j^x & \leq & x & \leq & u_j^x, \quad j = 1, \ldots, n.
\end{array}$$

The matrix

$$Q^o = \left[ \begin{array}{ccc} q_{11}^o & \cdots & q_{1n}^o \\ \vdots & \ldots & \vdots \\ q_{n1}^0 & \cdots & q_{nn}^o \end{array} \right]$$

must be symmetric positive semidefinite and the matrix

$$Q^i = \left[ \begin{array}{ccc} q_{11}^i & \cdots & q_{1n}^i \\ \vdots & \ldots & \vdots \\ q_{n1}^i & \cdots & q_{nn}^i \end{array} \right]$$

must be either symmetric negative semidefinite with the $i$th constraint

$$l_i^c \leq \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} q_{j,k}^i x_j x_k + \sum_{j=1}^{n} a_{i,j} x_j,$$

or $Q^i$ must be symmetric positive semidefinite with the $i$th constraint

$$\frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} q_{j,k}^i x_j x_k + \sum_{j=1}^{n} a_{i,j} x_j \leq u_i^c.$$

Note that the if the quadratic terms $Q^i$ are absent, the problem reduces to a standard quadratic optimization problem.

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

$$x_j \text{ integer-constrained for all } j \in | \tag{17.6}$$

where $x_j$ (and possibly $\overline{X}_j$) are the decision variables and all the other quantities are the parameters of the problem and they are presented below:

- Since $Q^o$ and $Q^i$ are symmetric, only the lower triangular part should be specified.
- The coefficients $c_j$ are coefficients for the linear term $c_j x_j$ in the objective.
- $c^f$ is a constant term in the objective, i.e., independent of all variables.
- The constraint matrix $A$ is given by

$$A = \left[ \begin{array}{ccc} a_{11} & \cdots & a_{1n} \\ \vdots & \ldots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{array} \right].$$

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

- The symmetric matrices $\overline{C}_j$ are coefficient matrices for the linear term $\mathrm{tr}(\overline{C}_j \overline{X}_j)$ in the objective for semidefinite problems. The matrices are specified in triplet format discarding zero elements, and since they are symmetric, only the lower triangular parts should be specified.
- The constraint matrices $\overline{A}_{ij}$ are symmetric matrices used in the constraints

$$l_i^c \leq \sum_{j=1}^{n} a_{ij} x_j + \sum_{j=1}^{p} \langle \overline{A}_{ij}, \overline{X}_j \rangle \leq u_i^c, \quad i = 1, \ldots, m,$$

  for semidefinite problems. The matrices are specifed in triplet format discard zero elements, and since they are symmetric only the lower triangulars should be specified.
  - $l^c$ specifies the lower bounds of the constraints.
  - $u^c$ specifies the upper bounds of the constraints.
  - $l^x$ specifies the lower bounds on the variables $x$.
  - $u^x$ specifies the upper bounds on the variables $x$.
- In conic problems, a partitioning of $x$ belongs to a set of free variables and quadratic cones. Let

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

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

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

  Next, define

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

  where $\mathcal{K}_t$ must have one of the following forms
- Free variables:

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

- Quadratic cones:

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

- Rotated quadratic cones:

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

The parameters of the optimization problem are stored using one or more subfields of the `prob` structure using the naming convention in Table 17.1.

Table 17.1: The relation between fields and problem parameters

| Field name | Type | Dimension | Optional | Problem parameter |
|---|---|---|---|---|
| qosubi | int | length(qoval) | Yes | $q_{ij}^o$ |
| qosubj | int | length(qoval) | Yes | $q_{ij}^o$ |
| qoval | double | length(qoval) | Yes | $q_{ij}^o$ |
| c | double | $n$ | Yes | $c_j$ |
| qcsubk | int | length(qcval) | Yes | $q_{ij}^p$ |
| qcsubi | int | length(qcval) | Yes | $q_{ij}^p$ |
| qcsubj | int | length(qcval) | Yes | $q_{ij}^p$ |
| qcval | double | length(qcval) | Yes | $q_{ij}^p$ |
| a | Sparse matrix | $mn$ | No | $a_{ij}$ |
| bardim | int | $p$ | Yes | $r_j$ |
| barc | MATLAB struct | | Yes | $\overline{C}_j$ |
| bara | MATLAB struct | | Yes | $\overline{A}_{ij}$ |
| blc | double | $m$ | Yes | $l_k^c$ |
| buc | double | $m$ | Yes | $u_k^c$ |
| blx | double | $n$ | Yes | $l_k^x$ |
| bux | double | $n$ | Yes | $u_k^x$ |
| ints | MATLAB struct | $\|\mathcal{J}\|$ | Yes | $\mathcal{J}$ |
| cones | MATLAB cell | $k$ | Yes | $\mathcal{K}$ |

In Table 17.1 all the parameters are listed with their corresponding type. The `int` type indicates that the field must contain an integer value, `double` indicates a real number. The relationship between $Q^o$ and $Q^p$ and the subfields of the `prob` structure is as follows:

- The quadratic terms in the objective:

$$q^o_{\mathtt{qosubi(t)},\mathtt{qoval(t)}} = \mathtt{qoval(t)}, \quad t = 1, 2, \ldots, \mathtt{length(qoval)}. \tag{17.7}$$

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

- The quadratic terms in the constraints:

$$q^{\mathtt{qcsubk(t)}}_{\mathtt{qcsubi(t)},\mathtt{qcsubj(t)}} = \mathtt{qcval(t)}, \quad t = 1, 2, \ldots, \mathtt{length(qcval)}. \tag{17.8}$$

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

## 17.2.2 Data Types and Structures

`prob`

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

**Fields**

- `names` (string) – A structure which contains the problem name, the name of the objective, and so forth.
- `qosubi` (int[]) – $i$ subscript for element $q_{ij}^o$ in $Q^o$. See (17.7).
- `qosubj` (int[]) – $j$ subscript for element $q_{ij}^o$ in $Q^o$. See (17.7).
- `qoval` (double[]) – Numerical value for element $q_{ij}^o$ in $Q^o$. See (17.7).
- `qcsubk` (int[]) – $k$ subscript for element $q_{ij}^p$ in $Q^p$. See (17.8)
- `qcsubi` (int[]) – $i$ subscript for element $q_{ij}^p$ in $Q^p$. See (17.8)

- qcsubj (double[]) – $j$ subscript for element $q_{ij}^p$ in $Q^p$. See (17.8)
- qcval (double[]) – Numerical value for element $q_{ij}^p$ in $Q^p$. See (17.8)
- c (double[]) – Linear term in the objective.
- a (double[][]) – The constraint matrix. It must be a **sparse matrix** having the number of rows and columns equivalent to the number of constraints and variables in the problem. This field should always be defined, even if the problem does not have any constraints. In that case a sparse matrix having zero rows and the correct number of columns is the appropriate definition of the field.
- blc (double[]) – Lower bounds of the constraints. $-\infty$ denotes an infinite lower bound. If the field is not defined or blc==[], then all the lower bounds are assumed to be equal to $-\infty$.
- bardim (int[]) – A list with the dimensions of the semidefinite variables.
- barc (barc) – A structure for specifying $\overline{C}_j$.
- bara (bara) – A structure for specifying $\overline{A}_{ij}$.
- buc (double[]) – Upper bounds of the constraints. $\infty$ denotes an infinite upper bound. If the field is not defined or buc==[], then all the upper bounds are assumed to be equal to $\infty$.
- blx (double[]) – Lower bounds on the variables. $-\infty$ denotes an infinite lower bound. If the field is not defined or blx==[], then all the lower bounds are assumed to be equal to $-\infty$.
- bux (double[]) – Upper bounds on the variables. $\infty$ denotes an infinite upper bound. If the field is not defined or bux==[], then all the upper bounds are assumed to be equal to $\infty$.
- ints (struct) – A structure which has the subfields
  - .sub A one-dimensional array containing the indexes of the integer-constrained variables. ints.sub is identical to the set $|$ in (17.6).
  - .pri A one dimensional array of the same length as ints.sub. The ints.pri(k) is the branching priority assigned to variable index ints.sub(k).
- cones (cones) – A structure defining the conic constraints (17.5).
- sol (solver_solutions) – A structure containing a guess on the optimal solution which some of the optimizers in **MOSEK** may exploit.
- primalrepair (struct) – A structure used for primal feasibility repair which can optimally contain either of the subfields:
  - .wlc Weights for lower bounds on constraints.
  - .wuc Weights for upper bounds on constraints.
  - .wlx Weights for lower bounds on variables.
  - .wlc Weights for upper bounds on variables.

  If either of the subfields is missing, it assumed to be a vector with value 1 of appropriate dimension.
- prisen (prisen) – A structure which has the subfields:

res

**Fields**

- sol (solver_solutions) – A structure holding available solutions (if any)

- `info` (struct) – A structure containing the task information database which contains various task related information such as the number of iterations used to solve the problem. However, this field is only defined if info appeared in the cmd command when *mosekopt* is invoked.
- `param` (list) – A structure which contain the complete **MOSEK** parameter database. However, this field is defined only if the param command is present in `cmd` when *mosekopt* is invoked.
- `prob` (prob) – Contains the problem data if the problem data was read from a file.

`names`

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

**Fields**

- `name` (string) – contains the problem name.
- `obj` (string) – contains the name of the objective.
- `con` (cell) – a cell array where `names.con{i}` contains the name of the $i$th constraint.
- `var` (cell) – a cell array where `names.var{j}` contains the name of the $j$th variable.
- `barvar` (cell) – a cell array where `names.barvar{j}` contains the name of the $j$th semidefinite variable.
- `cone` (cell) – a cell array where `names.cone{t}` contains the name of the $t$th conic constraint.

`cones`

A MATLAB structure representing details about cones.

For example the quadratic cone

$$x_5 \geq \sqrt{{x_3}^2 + {x_1}^2}$$

and rotated quadratic cone

$$2x_6x_4 \geq x_2^2 + x_7^2$$

would be specified using the two arrays

```
cones.type   = [0, 1];
cones.sub    = [5, 3, 1, 6, 4, 2, 7];
cones.subptr = [1, 4];
```

**Fields**

- `type` (list) – An array with the cone types for each cone; *"MSK_CT_QUAD"* or *"MSK_CT_RQUAD"*, indicating if the cone is a quadratic cone or a rotated quadratic cone.
- `sub` (int[]) – An array of variable indexes specifying which variables are fields of the cones. The array is a concatenation of index lists of all the cones.
- `subptr` (int[]) – An array of pointers into `cones.sub` indicating the beginning of the different cone index-sets.

`barc`

Together with field `bardim` this structure specifies the symmetric matrices $\overline{C}_j$ in the objective for semidefinite problems.

The symmetric matrices are specified in block-triplet format as

$$[\overline{C}_{\mathtt{barc.subj(t)}}]_{\mathtt{barc.subk(t)},\mathtt{barc.subl(t)}} = \mathtt{barc.val(t)}, \quad t = 1, 2, \ldots, \mathtt{length}(\mathtt{barc.subj}).$$

Only the lower triangular parts of $\overline{C}_j$ are specified, i.e., it is required that

$$\mathtt{barc.subk(t)} \geq \mathtt{barc.subl(t)}, \quad t = 1, 2, \ldots, \mathtt{length}(\mathtt{barc.subk}),$$

and that

$$1 \leq \mathtt{barc.subk(t)} \leq \mathtt{bardim}(\mathtt{barc.subj(t)}), \quad t = 1, 2, \ldots, \mathtt{length}(\mathtt{barc.subj}).,$$

All the structure fields must be arrays of the same length.

**Fields**

- `subj` (int[]) – Semidefinite variable indices $j$.
- `subk` (int[]) – Subscripts of nonzeros elements.
- `subl` (int[]) – Subscripts of nonzeros elements.
- `val` (double) – Numerical values.

`bara`

Together with the field `bardim` this structure specifies the symmetric matrices $\overline{A}_{ij}$ in the constraints of semidefinite problems.

The symmetric matrices are specified in block-triplet format as

$$[\overline{A}_{\mathtt{bara.subi(t)},\mathtt{bara.subj(t)}}]_{\mathtt{bara.subk(t)},\mathtt{bara.subl(t)}} = \mathtt{bara.val(t)}, \quad t = 1, 2, \ldots, \mathtt{length}(\mathtt{bara.subi}).$$

Only the lower triangular parts of $\overline{A}_{ij}$ are specified, i.e., it is required that

$$\mathtt{bara.subk(t)} \geq \mathtt{bara.subl(t)}, \quad t = 1, 2, \ldots, \mathtt{length}(\mathtt{bara.subk}),$$

and that

$$1 \leq \mathtt{bara.subk(t)} \leq \mathtt{bardim}(\mathtt{bara.subj(t)}), \quad t = 1, 2, \ldots, \mathtt{length}(\mathtt{bara.subj}),$$

**Fields**

- `subi` (int) – Constraint indices $i$.
- `subj` (int) – Semidefinite variable indices $j$.
- `subk` (int[]) – Subscripts of nonzeros elements.
- `subl` (int[]) – Subscripts of nonzeros elements.
- `val` (double[]) – Numerical values.

`solver_solutions`

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

**Fields**

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

`solution`

Stores information about a solution returned by the solve.

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

**Fields**

- `prosta` (prosta) – Problem status.
- `solsta` (solsta) – Solution status.
- `skc` (stakey) – Enumraint status keys.
- `skx` (stakey) – Variable status keys.
- `skn` (stakey) – Conic status keys.
- `xc` (double[]) – Constraint activities, i.e., $x_c = Ax$ where $x$ is the optimal solution.
- `xx` (double[]) – The optimal $x$ solution.
- `barx` (list) – The optimal solution of $\overline{X}_j, \quad j = 1, 2, \ldots,$ `length`(`bardim`).
- `bars` (list) – The optimal solution of $\overline{S}_j, \quad j = 1, 2, \ldots,$ `length`(`bardim`).
- `y` (double[]) – Identical to `sol.slc-sol.suc`.
- `slc` (double[]) – Dual solution corresponding to the lower constraint bounds.
- `suc` (double[]) – Dual solution corresponding to the upper constraint bounds.
- `slx` (double[]) – Dual solution corresponding to the lower variable bounds.
- `sux` (double[]) – Dual solution corresponding to the upper variable bounds.
- `snx` (double[]) – Dual solution corresponding to the conic constraint.
- `pobjval` (double) – The primal objective value.

`prisen`

Results of the primal sensitivity analysis.

**Fields**

- `cons` (cprisen) – Constraints shadow prices.
- `var` (vprisen) – Variable shadow prices
- `sub` (int[]) – Index of variables where coefficients are analysed for sensitivity.

`cprisen`

A structure holding information about constraint shadow prices.

**Fields**

- `lr_bl` (double) – Left value $\beta_1$ in the linearity interval for a lower bound.
- `rr_bl` (double) – Right value $\beta_2$ in the linearity interval for a lower bound.
- `ls_bl` (double) – Left shadow price $s_l$ for a lower bound.
- `rs_bl` (double) – Right shadow price $s_r$ for a lower bound.
- `lr_bu` (double) – Left value $\beta_1$ in the linearity interval for an upper bound.
- `rr_bu` (double) – Right value $\beta_2$ in the linearity interval for an upper bound.
- `ls_bu` (double) – Left shadow price $s_l$ for an upper bound.
- `rs_bu` (double) – Right shadow price $s_r$ for an upper bound.

`vprisen`

A structure holding information about variable shadow prices.

**Fields**

- `lr_bl` (double) – Left value $\beta_1$ in the linearity interval for a lower bound on a variable
- `rr_bl` (double) – Right value $\beta_2$ in the linearity interval for a lower bound on a variable
- `ls_bl` (double) – Left shadow price $s_l$ for a lower bound on a variable
- `rs_bl` (double) – Right shadow price $s_r$ for a lower bound on a variable
- `lr_bu` (double) – Left value $\beta_1$ in the linearity interval for an upper bound on a variable
- `rr_bu` (double) – Right value $\beta_2$ in the linearity interval for an upper bound on a variable
- `ls_bu` (double) – Left shadow price $s_l$ for an upper bound on a variable
- `rs_bu` (double) – Right shadow price $s_r$ for an upper bound on a variable.

`duasen`

Results of dual the sensitivity analysis.

**Fields**

- `lr_c` (double) – Left value $\beta_1$ in linearity interval for an objective coefficient
- `rr_c` (double) – Right value $\beta_2$ in linearity interval for an objective coefficient
- `ls_c` (double) – Left shadow price $s_l$ for an objective coefficient
- `rs_c` (double) – Right shadow price $s_r$ for an objective coefficient

`info`

`info` is a MATLAB structure containing a subfield for each item in the **MOSEK** optimization task database, e.g., the `info.dinfitem.bi_time` field specifies the amount of time spent in the basis identification in the last optimization. See *`dinfitem`* and *`iinfitem`* for all the items in the task information database are listed.

`symbcon`

A MATLAB structure containing a subfield for each **MOSEK** symbolic constant, e.g., the field `symbcon.dinfitem.bi_time` specifies the value of the symbolic constant *`"MSK_DINF_BI_TIME"`*. In Sec. 17.6 allthe symbolic constants are listed.

`callback`

A structure containing callback information (all subfields are optional).

**Fields**

- `loghandle` (struct) – A data structure or just `[]`.
- `log` (string) – The name of a user-defined function which must accept two input arguments, e.g.,

  ```
  function myfunc(handle,str)
  ```

  where `handle` will be identical to `callback.handle` when `myfunc` is called, and `str` is a string of text from the log file.
- `iterhandle` (struct) – A data structure or just `[]`.
- `iter` (string) – The name of a user-defined function which must accept three input arguments,

  ```
  function myfunc(handle,where,info)
  ```

where `handle` will be identical to `callback.iterhandle` when `myfunc` is called, `where` indicates the current progress of the colver and `info` is the current information database. See *info* for further details.

## 17.3 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*
- *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_MAX_TERMS_PER_LINE*
- *MSK_IPAR_OPF_WRITE_HEADER*
- *MSK_IPAR_OPF_WRITE_HINTS*
- *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_READ_DATA_COMPRESSED*
- *MSK_IPAR_READ_DATA_FORMAT*
- *MSK_IPAR_READ_DEBUG*
- *MSK_IPAR_READ_KEEP_FREE_CON*
- *MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU*
- *MSK_IPAR_READ_LP_QUOTED_NAMES*
- *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_DATA_COMPRESSED*
- *MSK_IPAR_WRITE_DATA_FORMAT*

- *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_LP_FULL_OBJ*
- *MSK_IPAR_WRITE_LP_LINE_WIDTH*
- *MSK_IPAR_WRITE_LP_QUOTED_NAMES*
- *MSK_IPAR_WRITE_LP_STRICT_FORMAT*
- *MSK_IPAR_WRITE_LP_TERMS_PER_LINE*
- *MSK_IPAR_WRITE_MPS_FORMAT*
- *MSK_IPAR_WRITE_MPS_INT*
- *MSK_IPAR_WRITE_PRECISION*
- *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_FILE_NAME*
- *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_NL_MERIT_BAL*
- *MSK_DPAR_INTPNT_NL_TOL_DFEAS*
- *MSK_DPAR_INTPNT_NL_TOL_MU_RED*
- *MSK_DPAR_INTPNT_NL_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_NL_TOL_PFEAS*
- *MSK_DPAR_INTPNT_NL_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_NL_TOL_REL_STEP*
- *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_METHOD*
- *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_INFEAS_ANA*
- *MSK_IPAR_LOG_INTPNT*
- *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_DISABLE_TERM_TIME*
- *MSK_DPAR_MIO_MAX_TIME*
- *MSK_DPAR_MIO_NEAR_TOL_ABS_GAP*
- *MSK_DPAR_MIO_NEAR_TOL_REL_GAP*
- *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_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_SELECTION_LEVEL*
- *MSK_IPAR_MIO_HEURISTIC_LEVEL*
- *MSK_IPAR_MIO_MAX_NUM_BRANCHES*
- *MSK_IPAR_MIO_MAX_NUM_RELAXS*
- *MSK_IPAR_MIO_MAX_NUM_SOLUTIONS*
- *MSK_IPAR_MIO_NODE_OPTIMIZER*
- *MSK_IPAR_MIO_NODE_SELECTION*
- *MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE*
- *MSK_IPAR_MIO_PROBING_LEVEL*
- *MSK_IPAR_MIO_RINS_MAX_NODES*
- *MSK_IPAR_MIO_ROOT_OPTIMIZER*
- *MSK_IPAR_MIO_ROOT_REPEAT_PRESOLVE_LEVEL*
- *MSK_IPAR_MIO_VB_DETECTION_LEVEL*

### Nonlinear convex method

- *MSK_DPAR_INTPNT_NL_MERIT_BAL*
- *MSK_DPAR_INTPNT_NL_TOL_DFEAS*
- *MSK_DPAR_INTPNT_NL_TOL_MU_RED*
- *MSK_DPAR_INTPNT_NL_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_NL_TOL_PFEAS*
- *MSK_DPAR_INTPNT_NL_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_NL_TOL_REL_STEP*
- *MSK_DPAR_INTPNT_TOL_INFEAS*
- *MSK_IPAR_CHECK_CONVEXITY*
- *MSK_IPAR_LOG_CHECK_CONVEXITY*

### 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_INFEAS_ANA*
- *MSK_IPAR_LOG_INTPNT*
- *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_INTPNT_MULTI_THREAD*
- *MSK_IPAR_LICENSE_WAIT*
- *MSK_IPAR_LOG_STORAGE*
- *MSK_IPAR_MIO_MT_USER_CB*

- *MSK_IPAR_MT_SPINCOUNT*
- *MSK_IPAR_NUM_THREADS*
- *MSK_IPAR_REMOVE_UNUSED_SOLUTIONS*
- *MSK_IPAR_TIMING_LEVEL*
- *MSK_SPAR_REMOTE_ACCESS_TOKEN*

### Presolve

- *MSK_DPAR_PRESOLVE_TOL_ABS_LINDEP*
- *MSK_DPAR_PRESOLVE_TOL_AIJ*
- *MSK_DPAR_PRESOLVE_TOL_REL_LINDEP*
- *MSK_DPAR_PRESOLVE_TOL_S*
- *MSK_DPAR_PRESOLVE_TOL_X*
- *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_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_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_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_NL_TOL_DFEAS*
- *MSK_DPAR_INTPNT_NL_TOL_MU_RED*
- *MSK_DPAR_INTPNT_NL_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_NL_TOL_PFEAS*
- *MSK_DPAR_INTPNT_NL_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_DISABLE_TERM_TIME*
- *MSK_DPAR_MIO_MAX_TIME*
- *MSK_DPAR_MIO_NEAR_TOL_REL_GAP*
- *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_SOLUTIONS*
- *MSK_IPAR_SIM_MAX_ITERATIONS*

#### Other

- *MSK_IPAR_COMPRESS_STATFILE*

## 17.4 Parameters (alphabetical list sorted by type)

- *Double parameters*
- *Integer parameters*
- *String parameters*

### 17.4.1 Double parameters

`dparam`
The enumeration type containing all 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]

**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]

**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]

**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]

**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:

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

**Default** 1e-10

**Accepted** [0; +inf]

**Groups** *Interior-point method*

MSK_DPAR_DATA_SYM_MAT_TOL

Absolute zero tolerance for elements in in suymmetric matrixes. 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]

**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]

**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]

**Groups** *Data check*

MSK_DPAR_DATA_TOL_AIJ

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

**Default** 1.0e-12

**Accepted** [1.0e-16; 1.0e-6]

**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]

**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]

**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]

**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]

**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]

**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]

**Groups** *Data check*

MSK_DPAR_DATA_TOL_QIJ

Absolute zero tolerance for elements in $Q$ matrices.

**Default** 1.0e-16

**Accepted** [0.0; +inf]

**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]

**Groups** *Data check*

MSK_DPAR_INTPNT_CO_TOL_DFEAS

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

**Default** 1.0e-8

**Accepted** [0.0; 1.0]

**Groups** *Interior-point method*, *Termination criteria*, *Conic interior-point method*

**See also** *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*

MSK_DPAR_INTPNT_CO_TOL_INFEAS

Controls when the conic interior-point optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

**Default** 1.0e-10

**Accepted** [0.0; 1.0]

**Groups** *Interior-point method*, *Termination criteria*, *Conic interior-point method*

MSK_DPAR_INTPNT_CO_TOL_MU_RED

Relative complementarity gap feasibility tolerance used by the conic interior-point optimizer.

**Default** 1.0e-8

**Accepted** [0.0; 1.0]

**Groups** *Interior-point method*, *Termination criteria*, *Conic interior-point method*

MSK_DPAR_INTPNT_CO_TOL_NEAR_REL

If **MOSEK** cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.

**Default** 1000

**Accepted** [1.0; +inf]

**Groups** *Interior-point method*, *Termination criteria*, *Conic interior-point method*

MSK_DPAR_INTPNT_CO_TOL_PFEAS

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

**Default** 1.0e-8

**Accepted** [0.0; 1.0]

**Groups** *Interior-point method*, *Termination criteria*, *Conic interior-point method*

**See also** *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*

MSK_DPAR_INTPNT_CO_TOL_REL_GAP

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

**Default** 1.0e-7

**Accepted** [0.0; 1.0]

**Groups** *Interior-point method*, *Termination criteria*, *Conic interior-point method*

**See also** *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*

MSK_DPAR_INTPNT_NL_MERIT_BAL

Controls if the complementarity and infeasibility is converging to zero at about equal rates.

**Default** 1.0e-4

**Accepted** [0.0; 0.99]

**Groups** *Interior-point method*, *Nonlinear convex method*

MSK_DPAR_INTPNT_NL_TOL_DFEAS

Dual feasibility tolerance used when a nonlinear model is solved.

**Default** 1.0e-8

**Accepted** [0.0; 1.0]

**Groups** *Interior-point method*, *Termination criteria*, *Nonlinear convex method*

MSK_DPAR_INTPNT_NL_TOL_MU_RED
Relative complementarity gap tolerance for the nonlinear solver.

**Default** 1.0e-12

**Accepted** [0.0; 1.0]

**Groups** *Interior-point method*, *Termination criteria*, *Nonlinear convex method*

MSK_DPAR_INTPNT_NL_TOL_NEAR_REL
If the **MOSEK** nonlinear interior-point optimizer cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.

**Default** 1000.0

**Accepted** [1.0; +inf]

**Groups** *Interior-point method*, *Termination criteria*, *Nonlinear convex method*

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

**Default** 1.0e-8

**Accepted** [0.0; 1.0]

**Groups** *Interior-point method*, *Termination criteria*, *Nonlinear convex method*

MSK_DPAR_INTPNT_NL_TOL_REL_GAP
Relative gap termination tolerance for nonlinear problems.

**Default** 1.0e-6

**Accepted** [1.0e-14; +inf]

**Groups** *Termination criteria*, *Interior-point method*, *Nonlinear convex method*

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

**Default** 0.995

**Accepted** [1.0e-4; 0.9999999]

**Groups** *Interior-point method*, *Nonlinear convex method*

MSK_DPAR_INTPNT_QO_TOL_DFEAS
Dual feasibility tolerance used when the interior-point optimizer is applied to a quadratic optimization problem..

**Default** 1.0e-8

**Accepted** [0.0; 1.0]

**Groups** *Interior-point method*, *Termination criteria*

**See also** *MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*

MSK_DPAR_INTPNT_QO_TOL_INFEAS
Controls when the conic interior-point optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

**Default** 1.0e-10

**Accepted** [0.0; 1.0]

**Groups** *Interior-point method*, *Termination criteria*

MSK_DPAR_INTPNT_QO_TOL_MU_RED
Relative complementarity gap feasibility tolerance used when interior-point optimizer is applied to a quadratic optimization problem.

**Default** 1.0e-8

**Accepted** [0.0; 1.0]

**Groups** *Interior-point method*, *Termination criteria*

MSK_DPAR_INTPNT_QO_TOL_NEAR_REL

If **MOSEK** cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.

**Default** 1000

**Accepted** [1.0; +inf]

**Groups** *Interior-point method*, *Termination criteria*

MSK_DPAR_INTPNT_QO_TOL_PFEAS

Primal feasibility tolerance used when the interior-point optimizer is applied to a quadratic optimization problem.

**Default** 1.0e-8

**Accepted** [0.0; 1.0]

**Groups** *Interior-point method*, *Termination criteria*

**See also** *MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*

MSK_DPAR_INTPNT_QO_TOL_REL_GAP

Relative gap termination tolerance used when the interior-point optimizer is applied to a quadratic optimization problem.

**Default** 1.0e-8

**Accepted** [0.0; 1.0]

**Groups** *Interior-point method*, *Termination criteria*

**See also** *MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*

MSK_DPAR_INTPNT_TOL_DFEAS

Dual feasibility tolerance used for linear optimization problems.

**Default** 1.0e-8

**Accepted** [0.0; 1.0]

**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]

**Groups** *Interior-point method*

MSK_DPAR_INTPNT_TOL_INFEAS

Controls when the optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

**Default** 1.0e-10

**Accepted** [0.0; 1.0]

**Groups** *Interior-point method*, *Termination criteria*, *Nonlinear convex method*

MSK_DPAR_INTPNT_TOL_MU_RED

Relative complementarity gap tolerance for linear problems.

**Default** 1.0e-16

**Accepted** [0.0; 1.0]

**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 is followed very closely. On numerical unstable problems it may be worthwhile to increase this parameter.

**Default** 1.0e-8

**Accepted** [0.0; 0.9999]

**Groups** *Interior-point method*

MSK_DPAR_INTPNT_TOL_PFEAS
Primal feasibility tolerance used for linear optimization problems.

**Default** 1.0e-8

**Accepted** [0.0; 1.0]

**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]

**Groups** *Interior-point method*

MSK_DPAR_INTPNT_TOL_REL_GAP
Relative gap termination tolerance for linear problems.

**Default** 1.0e-8

**Accepted** [1.0e-14; +inf]

**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]

**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 stop.

**Default** 1.0e-6

**Accepted** [0.0; 1.0]

**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]

**Groups** *Termination criteria*

**See also** *MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH*

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]

**Groups** *Termination criteria*

MSK_DPAR_MIO_DISABLE_TERM_TIME

This parameter specifies the number of seconds $n$ during which the termination criteria governed by

- *MSK_IPAR_MIO_MAX_NUM_RELAXS*
- *MSK_IPAR_MIO_MAX_NUM_BRANCHES*
- *MSK_DPAR_MIO_NEAR_TOL_ABS_GAP*
- *MSK_DPAR_MIO_NEAR_TOL_REL_GAP*

is disabled since the beginning of the optimization.

A negative value is identical to infinity i.e. the termination criteria are never checked.

**Default** -1.0

**Accepted** [-inf; +inf]

**Groups** *Mixed-integer optimization*, *Termination criteria*

**See also** *MSK_IPAR_MIO_MAX_NUM_RELAXS*, *MSK_IPAR_MIO_MAX_NUM_BRANCHES*, *MSK_DPAR_MIO_NEAR_TOL_ABS_GAP*, *MSK_DPAR_MIO_NEAR_TOL_REL_GAP*

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]

**Groups** *Mixed-integer optimization*, *Termination criteria*

MSK_DPAR_MIO_NEAR_TOL_ABS_GAP

Relaxed absolute optimality tolerance employed by the mixed-integer optimizer. This termination criteria is delayed. See *MSK_DPAR_MIO_DISABLE_TERM_TIME* for details.

**Default** 0.0

**Accepted** [0.0; +inf]

**Groups** *Mixed-integer optimization*

**See also** *MSK_DPAR_MIO_DISABLE_TERM_TIME*

MSK_DPAR_MIO_NEAR_TOL_REL_GAP

The mixed-integer optimizer is terminated when this tolerance is satisfied. This termination criteria is delayed. See *MSK_DPAR_MIO_DISABLE_TERM_TIME* for details.

**Default** 1.0e-3

**Accepted** [0.0; +inf]

**Groups** *Mixed-integer optimization*, *Termination criteria*

**See also** *MSK_DPAR_MIO_DISABLE_TERM_TIME*

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]

**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]

**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]

**Groups** *Mixed-integer optimization*

MSK_DPAR_MIO_TOL_FEAS
Feasibility tolerance for mixed integer solver.

**Default** 1.0e-6

**Accepted** [1e-9; 1e-3]

**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]

**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]

**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]

**Groups** *Termination criteria*

MSK_DPAR_PRESOLVE_TOL_ABS_LINDEP
Absolute tolerance employed by the linear dependency checker.

**Default** 1.0e-6

**Accepted** [0.0; +inf]

**Groups** *Presolve*

MSK_DPAR_PRESOLVE_TOL_AIJ

Absolute zero tolerance employed for $a_{ij}$ in the presolve.

**Default** 1.0e-12

**Accepted** [1.0e-15; +inf]

**Groups** *Presolve*

MSK_DPAR_PRESOLVE_TOL_REL_LINDEP

Relative tolerance employed by the linear dependency checker.

**Default** 1.0e-10

**Accepted** [0.0; +inf]

**Groups** *Presolve*

MSK_DPAR_PRESOLVE_TOL_S

Absolute zero tolerance employed for $s_i$ in the presolve.

**Default** 1.0e-8

**Accepted** [0.0; +inf]

**Groups** *Presolve*

MSK_DPAR_PRESOLVE_TOL_X

Absolute zero tolerance employed for $x_j$ in the presolve.

**Default** 1.0e-8

**Accepted** [0.0; +inf]

**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]

**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]

**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]

**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]

**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]

**Groups** *Termination criteria*

**See also** *MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH*

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 $\infty$.

**Default** 0.5e30

**Accepted** [-inf; +inf]

**Groups** *Termination criteria*

### 17.4.2 Integer parameters

iparam
The enumeration type containing all integer parameters.

MSK_IPAR_ANA_SOL_BASIS
Controls whether the basis matrix is analyzed in solution analyzer.

**Default** *"ON"*

**Accepted** *"ON"*, *"OFF"*

**Groups** *Analysis*

MSK_IPAR_ANA_SOL_PRINT_VIOLATED
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"*

**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"*

**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"*

**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"*

**Groups** *Simplex optimizer*

MSK_IPAR_BI_CLEAN_OPTIMIZER

Controls which simplex optimizer is used in the clean-up phase.

**Default** *"FREE"*

**Accepted** *"FREE"*, *"INTPNT"*, *"CONIC"*, *"PRIMAL_SIMPLEX"*, *"DUAL_SIMPLEX"*, *"FREE_SIMPLEX"*, *"MIXED_INT"*

**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"*

**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"*

**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]

**Groups** *Basis identification*, *Termination criteria*

MSK_IPAR_CACHE_LICENSE

Specifies if the license is kept checked out for the lifetime of the mosek environment (*"MSK_ON"*) or returned to the server immediately after the optimization (*"MSK_OFF"*).

By default the license is checked out for the lifetime of the **MOSEK** environment by the first call to the optimizer.

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

**Default** *"ON"*

**Accepted** *"ON"*, *"OFF"*

**Groups** *License manager*

MSK_IPAR_CHECK_CONVEXITY
Specify the level of convexity check on quadratic problems.

**Default** *"FULL"*

**Accepted** *"NONE"*, *"SIMPLE"*, *"FULL"*

**Groups** *Data check*, *Nonlinear convex method*

MSK_IPAR_COMPRESS_STATFILE
Control compression of stat files.

**Default** *"ON"*

**Accepted** *"ON"*, *"OFF"*

MSK_IPAR_INFEAS_GENERIC_NAMES
Controls whether generic names are used when an infeasible subproblem is created.

**Default** *"OFF"*

**Accepted** *"ON"*, *"OFF"*

**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"*

**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"*

**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]

**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"*, *"RESERVERED"*

**Groups** *Interior-point method*, *Basis identification*

**See also** *MSK_IPAR_BI_IGNORE_MAX_ITER*, *MSK_IPAR_BI_IGNORE_NUM_ERROR*, *MSK_IPAR_BI_MAX_ITERATIONS*, *MSK_IPAR_BI_CLEAN_OPTIMIZER*

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.

**Groups** *Interior-point method*

MSK_IPAR_INTPNT_HOTSTART
Currently not in use.

**Default** *"NONE"*

**Accepted** *"NONE"*, *"PRIMAL"*, *"DUAL"*, *"PRIMAL_DUAL"*

**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; +inf]

**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; +inf]

**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** [-inf; +inf]

**Groups** *Interior-point method*

MSK_IPAR_INTPNT_MULTI_THREAD
Controls whether the interior-point optimizers are allowed to employ multiple threads if more threads is available.

**Default** *"ON"*

**Accepted** *"ON"*, *"OFF"*

**Groups** *Overall system*

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; +inf]

**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"*

**Groups** *Interior-point method*

MSK_IPAR_INTPNT_REGULARIZATION_USE
Controls whether regularization is allowed.

**Default** *"ON"*

**Accepted** *"ON"*, *"OFF"*

**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"*, *"MODERATE"*, *"AGGRESSIVE"*

**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"*

**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"*

**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"*

**Groups** *License manager*

MSK_IPAR_LICENSE_PAUSE_TIME
If *MSK_IPAR_LICENSE_WAIT* = *"MSK_ON"* and no license is available, then **MOSEK** sleeps a number of milliseconds between each check of whether a license has become free.

**Default** 100

**Accepted** [0; 1000000]

**Groups** *License manager*

MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS
Controls whether license features expire warnings are suppressed.

**Default** *"OFF"*

**Accepted** *"ON"*, *"OFF"*

**Groups** *License manager*, *Output information*

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

**Default** 7

**Accepted** [0; +inf]

**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"*

**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]

**Groups** *Output information*, *Logging*

**See also** *MSK_IPAR_LOG_CUT_SECOND_OPT*

`MSK_IPAR_LOG_ANA_PRO`

Controls amount of output from the problem analyzer.

**Default** 1

**Accepted** [0; +inf]

**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]

**Groups** *Basis identification*, *Output information*, *Logging*

`MSK_IPAR_LOG_BI_FREQ`

Controls how frequent the optimizer outputs information about the basis identification and how frequent the user-defined callback function is called.

**Default** 2500

**Accepted** [0; +inf]

**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]

**Groups** *Data check*, *Nonlinear convex method*

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]

**Groups** *Output information*, *Logging*

**See also** *MSK_IPAR_LOG*, *MSK_IPAR_LOG_INTPNT*, *MSK_IPAR_LOG_MIO*, *MSK_IPAR_LOG_SIM*

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

**Default** 0

**Accepted** [0; +inf]

**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]

**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]

**Groups** *Data input/output*, *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]

**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]

**Groups** *Interior-point method*, *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]

**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]

**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]

**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]

**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]

**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]

**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]

**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]

**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]

**Groups** *Simplex optimizer*, *Output information*, *Logging*

MSK_IPAR_LOG_SIM_MINOR
Currently not in use.

**Default** 1

**Accepted** [0; +inf]

**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]

**Groups** *Output information*, *Overall system*, *Logging*

MSK_IPAR_MAX_NUM_WARNINGS
Each warning is shown a limit number times controlled by this parameter. A negative value is identical to infinite number of times.

**Default** 10

**Accepted** [-inf; +inf]

**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"*

**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"*

**Groups** *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_CLIQUE
Controls whether clique cuts should be generated.

**Default** *"ON"*

**Accepted**

- *"ON"*: Turns generation of this cut class on.
- *"OFF"*: Turns generation of this cut class off.

**Groups** *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_CMIR
Controls whether mixed integer rounding cuts should be generated.

**Default** *"ON"*

**Accepted**

- *"ON"*: Turns generation of this cut class on.
- *"OFF"*: Turns generation of this cut class off.

**Groups** *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_GMI
Controls whether GMI cuts should be generated.

**Default** *"ON"*

**Accepted**

- *"ON"*: Turns generation of this cut class on.
- *"OFF"*: Turns generation of this cut class off.

**Groups** *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_IMPLIED_BOUND
Controls whether implied bound cuts should be generated.

**Default** *"OFF"*

**Accepted**

- *"ON"*: Turns generation of this cut class on.
- *"OFF"*: Turns generation of this cut class off.

**Groups** *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_KNAPSACK_COVER
Controls whether knapsack cover cuts should be generated.

**Default** *"OFF"*

**Accepted**

- *"ON"*: Turns generation of this cut class on.
- *"OFF"*: Turns generation of this cut class off.

**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]

**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]

**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]

**Groups** *Mixed-integer optimization*, *Termination criteria*

**See also** *MSK_DPAR_MIO_DISABLE_TERM_TIME*

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]

**Groups** *Mixed-integer optimization*

**See also** *MSK_DPAR_MIO_DISABLE_TERM_TIME*

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]

**Groups** *Mixed-integer optimization*, *Termination criteria*

**See also** *MSK_DPAR_MIO_DISABLE_TERM_TIME*

MSK_IPAR_MIO_MODE

Controls whether the optimizer includes the integer restrictions when solving a (mixed) integer optimization problem.

**Default** *"SATISFIED"*

**Accepted** *"IGNORED"*, *"SATISFIED"*

**Groups** *Overall solver*

MSK_IPAR_MIO_MT_USER_CB

If true user callbacks are called from each thread used by mixed-integer optimizer. Otherwise it is only called from a single thread.

**Default** *"OFF"*

**Accepted** *"ON"*, *"OFF"*

**Groups** *Overall system*

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"*

**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"*, *"WORST"*, *"HYBRID"*, *"PSEUDO"*

**Groups** *Mixed-integer optimization*

MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE
Enables or disables perspective reformulation in presolve.

**Default** *"ON"*

**Accepted** *"ON"*, *"OFF"*

**Groups** *Mixed-integer optimization*

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]

**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]

**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"*

**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]

**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]

**Groups** *Mixed-integer optimization*

MSK_IPAR_MT_SPINCOUNT

Set the number of iterations to spin before sleeping.

**Default** 0

**Accepted** [0; 1000000000]

**Groups** *Overall system*

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]

**Groups** *Overall system*

MSK_IPAR_OPF_MAX_TERMS_PER_LINE

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

**Default** 5

**Accepted** [0; +inf]

**Groups** *Data input/output*

MSK_IPAR_OPF_WRITE_HEADER

Write a text header with date and **MOSEK** version in an OPF file.

**Default** *"ON"*

**Accepted** *"ON"*, *"OFF"*

**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"*

**Groups** *Data input/output*

MSK_IPAR_OPF_WRITE_PARAMETERS

Write a parameter section in an OPF file.

**Default** *"OFF"*

**Accepted** *"ON"*, *"OFF"*

**Groups** *Data input/output*

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

**Default** *"ON"*

**Accepted** *"ON"*, *"OFF"*

**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"*

**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"*

**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"*

**Groups** *Data input/output*

MSK_IPAR_OPF_WRITE_SOLUTIONS
Enable inclusion of solutions in the OPF files.

**Default** *"OFF"*

**Accepted** *"ON"*, *"OFF"*

**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"*

**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"*

**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"*

**Groups** *Data input/output*

MSK_IPAR_PRESOLVE_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]

**Groups** *Presolve*

MSK_IPAR_PRESOLVE_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]

**Groups** *Presolve*

MSK_IPAR_PRESOLVE_LEVEL
Currently not used.

**Default** -1

**Accepted** [-inf; +inf]

**Groups** *Overall solver*, *Presolve*

MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH
The linear dependency check is potentially computationally expensive.

**Default** 100

**Accepted** [-inf; +inf]

**Groups** *Presolve*

MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH
The linear dependency check is potentially computationally expensive.

**Default** 100

**Accepted** [-inf; +inf]

**Groups** *Presolve*

MSK_IPAR_PRESOLVE_LINDEP_USE
Controls whether the linear constraints are checked for linear dependencies.

**Default** *"ON"*

**Accepted**

- *"ON"*: Turns the linear dependency check on.
- *"OFF"*: Turns the linear dependency check off.

**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]

**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"*

**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"*

**Groups** *Overall solver*

MSK_IPAR_READ_DATA_COMPRESSED
If this option is turned on,it is assumed that the data file is compressed.

**Default** *"FREE"*

**Accepted** *"NONE"*, *"FREE"*, *"GZIP"*

**Groups** *Data input/output*

MSK_IPAR_READ_DATA_FORMAT
Format of the data file to be read.

**Default** *"EXTENSION"*

**Accepted** *"EXTENSION"*, *"MPS"*, *"LP"*, *"OP"*, *"XML"*, *"FREE_MPS"*, *"TASK"*, *"CB"*, *"JSON_TASK"*

**Groups** *Data input/output*

MSK_IPAR_READ_DEBUG
Turns on additional debugging information when reading files.

**Default** *"OFF"*

**Accepted** *"ON"*, *"OFF"*

**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.

**Groups** *Data input/output*

MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU
If this option is turned on, **MOSEK** will drop variables that are defined for the first time in the bounds section.

**Default** *"OFF"*

**Accepted** *"ON"*, *"OFF"*

**Groups** *Data input/output*

MSK_IPAR_READ_LP_QUOTED_NAMES
If a name is in quotes when reading an LP file, the quotes will be removed.

**Default** *"ON"*

**Accepted** *"ON"*, *"OFF"*

**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"*

**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]

**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"*

**Groups** *Data input/output*

MSK_IPAR_REMOVE_UNUSED_SOLUTIONS
Removes unsued solutions before the optimization is performed.

**Default** *"OFF"*

**Accepted** *"ON"*, *"OFF"*

**Groups** *Overall system*

MSK_IPAR_SENSITIVITY_ALL
Not applicable.

**Default** *"OFF"*

**Accepted** *"ON"*, *"OFF"*

**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"*

**Groups** *Overall solver*, *Simplex optimizer*

MSK_IPAR_SENSITIVITY_TYPE
Controls which type of sensitivity analysis is to be performed.

**Default** *"BASIS"*

**Accepted** *"BASIS"*, *"OPTIMAL_PARTITION"*

**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"*

**Groups** *Simplex optimizer*

MSK_IPAR_SIM_DEGEN

Controls how aggressively degeneration is handled.

**Default** *"FREE"*

**Accepted** *"NONE"*, *"FREE"*, *"AGGRESSIVE"*, *"MODERATE"*, *"MINIMUM"*

**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 $(100-x)$ mod $f_v$ entries, where $f_v$ is the number of fixed variables.

**Default** 90

**Accepted** [0; +inf]

**Groups** *Dual simplex*

MSK_IPAR_SIM_DUAL_PHASEONE_METHOD

An experimental feature.

**Default** 0

**Accepted** [0; 10]

**Groups** *Simplex optimizer*

MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION

The dual simplex optimizer can use a so-called restricted selection/pricing strategy to chooses the outgoing variable. Hence, if restricted selection is applied, then the dual simplex optimizer first choose a subset of all the potential outgoing variables. Next, for some time it will choose the outgoing variable only among the subset. From time to time the subset is redefined.

A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.

**Default** 50

**Accepted** [0; 100]

**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"*

**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"*

**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"*

**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.

**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]

**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]

**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"*

**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]

**Groups** *Primal simplex*

MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD
An experimental feature.

**Default** 0

**Accepted** [0; 10]

**Groups** *Simplex optimizer*

MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION
The primal simplex optimizer can use a so-called restricted selection/pricing strategy to chooses the outgoing variable. Hence, if restricted selection is applied, then the primal simplex optimizer

first choose a subset of all the potential incoming variables. Next, for some time it will choose the incoming variable only among the subset. From time to time the subset is redefined.

A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.

**Default** 50

**Accepted** [0; 100]

**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"*

**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]

**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"*

**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"*

**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"*, *"MODERATE"*, *"AGGRESSIVE"*

**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"*

**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"*

**Groups** *Simplex optimizer*

MSK_IPAR_SIM_STABILITY_PRIORITY

Controls how high priority the numerical stability should be given.

**Default** 50

**Accepted** [0; 100]

**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"*

**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"*

**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"*

**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 much be specified. A negative value implies that no name contain blanks.

**Default** -1

**Accepted** [-inf; +inf]

**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]

**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"*

**Groups** *Progress callback*, *Overall solver*

MSK_IPAR_TIMING_LEVEL
Controls the amount of timing performed inside **MOSEK**.

**Default** 1

**Accepted** [0; +inf]

**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"*

**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"*

**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"*

**Groups** *Data input/output*, *Solution input/output*

MSK_IPAR_WRITE_DATA_COMPRESSED
Controls whether the data file is compressed while it is written. 0 means no compression while higher values mean more compression.

**Default** 0

**Accepted** [0; +inf]

**Groups** *Data input/output*

MSK_IPAR_WRITE_DATA_FORMAT
Controls the file format when writing task data to a file.

**Default** *"EXTENSION"*

**Accepted** *"EXTENSION"*, *"MPS"*, *"LP"*, *"OP"*, *"XML"*, *"FREE_MPS"*, *"TASK"*, *"CB"*, *"JSON_TASK"*

**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"*

**Groups** *Data input/output*

MSK_IPAR_WRITE_FREE_CON
Controls whether the free constraints are written to the data file.

**Default** *"ON"*

**Accepted**

- *"ON"*: The free constraints are written.
- *"OFF"*: The free constraints are discarded.

**Groups** *Data input/output*

MSK_IPAR_WRITE_GENERIC_NAMES
Controls whether the generic names or user-defined names are used in the data file.

**Default** *"OFF"*

**Accepted**

- *"ON"*: Generic names are used.
- *"OFF"*: Generic names are not used.

**Groups** *Data input/output*

MSK_IPAR_WRITE_GENERIC_NAMES_IO
Index origin used in generic names.

**Default** 1

**Accepted** [0; +inf]

**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 the written to the current output file format.

**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"*

**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"*

**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"*

**Groups** *Data input/output*, *Solution 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"*

**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]

**Groups** *Data input/output*

MSK_IPAR_WRITE_LP_QUOTED_NAMES
If this option is turned on, then **MOSEK** will quote invalid LP names when writing an LP file.

**Default** *"ON"*

**Accepted** *"ON"*, *"OFF"*

**Groups** *Data input/output*

MSK_IPAR_WRITE_LP_STRICT_FORMAT
Controls whether LP output files satisfy the LP format strictly.

**Default** *"OFF"*

**Accepted** *"ON"*, *"OFF"*

**Groups** *Data input/output*

MSK_IPAR_WRITE_LP_TERMS_PER_LINE
Maximum number of terms on a single line in an LP file written by **MOSEK**. 0 means unlimited.

**Default** 10

**Accepted** [0; +inf]

**Groups** *Data input/output*

MSK_IPAR_WRITE_MPS_FORMAT
Controls in which format the MPS is written.

**Default** *"FREE"*

**Accepted** *"STRICT"*, *"RELAXED"*, *"FREE"*, *"CPLEX"*

**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"*: Marker records are written.
- *"OFF"*: Marker records are not written.

**Groups** *Data input/output*

MSK_IPAR_WRITE_PRECISION
Controls the precision with which `double` numbers are printed in the MPS data file. In general it is not worthwhile to use a value higher than 15.

**Default** 15

**Accepted** [0; +inf]

**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"*

**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"*

**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"*

**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"*

**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"*

**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"*

**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"*

**Groups** *Data input/output*

### 17.4.3 String parameters

sparam
The enumeration type containing all string parameters.

MSK_SPAR_BAS_SOL_FILE_NAME
Name of the `bas` solution file.

**Accepted** Any valid file name.

**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.

**Groups** *Data input/output*

`MSK_SPAR_DEBUG_FILE_NAME`
**MOSEK** debug file.

**Accepted** Any valid file name.

**Groups** *Data input/output*

`MSK_SPAR_INT_SOL_FILE_NAME`
Name of the `int` solution file.

**Accepted** Any valid file name.

**Groups** *Data input/output*, *Solution input/output*

`MSK_SPAR_ITR_SOL_FILE_NAME`
Name of the `itr` solution file.

**Accepted** Any valid file name.

**Groups** *Data input/output*, *Solution input/output*

`MSK_SPAR_MIO_DEBUG_STRING`
For internal debugging purposes.

**Accepted** Any valid string.

**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.

**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.

**Groups** *Data input/output*

`MSK_SPAR_PARAM_WRITE_FILE_NAME`
The parameter database is written to this file.

**Accepted** Any valid file name.

**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.

**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.

**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.

**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.

**Groups** *Data input/output*

`MSK_SPAR_REMOTE_ACCESS_TOKEN`
An access token used to submit tasks to a remote **MOSEK** server. An access token is a random 32-byte string encoded in base64, i.e. it is a 44 character ASCII string.

**Accepted** Any valid string.

**Groups** *Overall system*

`MSK_SPAR_SENSITIVITY_FILE_NAME`
If defined, **MOSEK** reads this file as a sensitivity analysis data file specifying the type of analysis to be done.

**Accepted** Any valid string.

**Groups** *Data input/output*

`MSK_SPAR_SENSITIVITY_RES_FILE_NAME`

**Accepted** Any valid string.

**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]>=blc[i]+0.5` should be listed. An empty filter means that no filter is applied.

**Accepted** Any valid filter.

**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]<=buc[i]-0.5` should be listed. An empty filter means that no filter is applied.

**Accepted** Any valid filter.

**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]>=0.5` should be listed, whereas "+0.5" means that all constraints having `xx[j]>=blx[j]+0.5` should be listed. An empty filter means no filter is applied.

**Accepted** Any valid filter.

**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]<=bux[j]-0.5` should be listed. An empty filter means no filter is applied.

**Accepted** Any valid file name.

**Groups** *Data input/output*, *Solution input/output*

`MSK_SPAR_STAT_FILE_NAME`

Statistics file name.

**Accepted** Any valid file name.

**Groups** *Data input/output*

`MSK_SPAR_STAT_KEY`

Key used when writing the summary file.

**Accepted** Any valid string.

**Groups** *Data input/output*

`MSK_SPAR_STAT_NAME`

Name used when writing the statistics file.

**Accepted** Any valid XML string.

**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.

**Groups** *Data input/output*

## 17.5 Response codes

- *Termination*
- *Warnings*
- *Errors*

`rescode`

The enumeration type containing all response codes.

### 17.5.1 Termination

`"MSK_RES_OK"`

No error occurred.

`"MSK_RES_TRM_MAX_ITERATIONS"`

The optimizer terminated at the maximum number of iterations.

`"MSK_RES_TRM_MAX_TIME"`

The optimizer terminated at the maximum amount of time.

`"MSK_RES_TRM_OBJECTIVE_RANGE"`

The optimizer terminated with an objective value outside the objective range.

"MSK_RES_TRM_MIO_NEAR_REL_GAP"
The mixed-integer optimizer terminated as the delayed near optimal relative gap tolerance was satisfied.

"MSK_RES_TRM_MIO_NEAR_ABS_GAP"
The mixed-integer optimizer terminated as the delayed near optimal absolute gap tolerance was satisfied.

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

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

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

"MSK_RES_TRM_STALL"
The optimizer is terminated due to slow progress.

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

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

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

"MSK_RES_TRM_USER_CALLBACK"
The optimizer terminated due to the return of the user-defined callback function.

"MSK_RES_TRM_MAX_NUM_SETBACKS"
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"
The optimizer terminated due to numerical problems.

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

"MSK_RES_TRM_INTERNAL_STOP"
The optimizer terminated for internal reasons. Please contact **MOSEK** support.

### 17.5.2 Warnings

"MSK_RES_WRN_OPEN_PARAM_FILE"
The parameter file could not be opened.

"MSK_RES_WRN_LARGE_BOUND"
A numerically large bound value is specified.

"MSK_RES_WRN_LARGE_LO_BOUND"
A numerically large lower bound value is specified.

"MSK_RES_WRN_LARGE_UP_BOUND"
A numerically large upper bound value is specified.

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

"MSK_RES_WRN_LARGE_CJ"
A numerically large value is specified for one $c_j$.

"MSK_RES_WRN_LARGE_AIJ"
A numerically large value is specified for an $a_{i,j}$ element in $A$. The parameter *MSK_DPAR_DATA_TOL_AIJ_LARGE* controls when an $a_{i,j}$ is considered large.

"MSK_RES_WRN_ZERO_AIJ"
One or more zero elements are specified in A.

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

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

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

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

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

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

"MSK_RES_WRN_LP_DROP_VARIABLE"
Ignored a variable because the variable was not previously defined. Usually this implies that a variable appears in the bound section but not in the objective or the constraints.

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

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

"MSK_RES_WRN_IGNORE_INTEGER"
Ignored integer constraints.

"MSK_RES_WRN_NO_GLOBAL_OPTIMIZER"
No global optimizer is available.

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

"MSK_RES_WRN_SOL_FILTER"
Invalid solution filter is specified.

"MSK_RES_WRN_UNDEF_SOL_FILE_NAME"
Undefined name occurred in a solution.

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

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

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

"MSK_RES_WRN_TOO_MANY_BASIS_VARS"
A basis with too many variables has been specified.

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

"MSK_RES_WRN_LICENSE_EXPIRE"
The license expires.

"MSK_RES_WRN_LICENSE_SERVER"
The license server is not responding.

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

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

"MSK_RES_WRN_LICENSE_FEATURE_EXPIRE"
The license expires.

"MSK_RES_WRN_PARAM_NAME_DOU"
The parameter name is not recognized as a double parameter.

"MSK_RES_WRN_PARAM_NAME_INT"
The parameter name is not recognized as a integer parameter.

"MSK_RES_WRN_PARAM_NAME_STR"
The parameter name is not recognized as a string parameter.

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

"MSK_RES_WRN_PARAM_IGNORED_CMIO"
A parameter was ignored by the conic mixed integer optimizer.

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

"MSK_RES_WRN_PRESOLVE_OUTOFSPACE"
The presolve is incomplete due to lack of space.

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

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

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

"MSK_RES_WRN_CONSTRUCT_INVALID_SOL_ITG"
The initial value for one or more of the integer variables is not feasible.

"MSK_RES_WRN_CONSTRUCT_NO_SOL_ITG"
The construct solution requires an integer solution.

"MSK_RES_WRN_DUPLICATE_CONSTRAINT_NAMES"
Two constraint names are identical.

"MSK_RES_WRN_DUPLICATE_VARIABLE_NAMES"
Two variable names are identical.

"MSK_RES_WRN_DUPLICATE_BARVARIABLE_NAMES"
Two barvariable names are identical.

"MSK_RES_WRN_DUPLICATE_CONE_NAMES"
Two cone names are identical.

"MSK_RES_WRN_ANA_LARGE_BOUNDS"
This warning is issued by the problem analyzer, if one or more constraint or variable bounds are very large. One should consider omitting these bounds entirely by setting them to +inf or -inf.

"MSK_RES_WRN_ANA_C_ZERO"
This warning is issued by the problem analyzer, if the coefficients in the linear part of the objective are all zero.

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

"MSK_RES_WRN_ANA_CLOSE_BOUNDS"
This warning is issued by problem analyzer, if ranged constraints or variables with very close upper and lower bounds are detected. One should consider treating such constraints as equalities and such variables as constants.

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

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

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

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

"MSK_RES_WRN_SYM_MAT_LARGE"
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.

### 17.5.3 Errors

"MSK_RES_ERR_LICENSE"
Invalid license.

"MSK_RES_ERR_LICENSE_EXPIRED"
The license has expired.

"MSK_RES_ERR_LICENSE_VERSION"
The license is valid for another version of **MOSEK**.

"MSK_RES_ERR_SIZE_LICENSE"
The problem is bigger than the license.

"MSK_RES_ERR_PROB_LICENSE"
The software is not licensed to solve the problem.

"MSK_RES_ERR_FILE_LICENSE"
Invalid license file.

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

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

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

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

"MSK_RES_ERR_OPTIMIZER_LICENSE"
The optimizer required is not licensed.

"MSK_RES_ERR_FLEXLM"
The FLEXlm license manager reported an error.

"MSK_RES_ERR_LICENSE_SERVER"
The license server is not responding.

"MSK_RES_ERR_LICENSE_MAX"
Maximum number of licenses is reached.

"MSK_RES_ERR_LICENSE_MOSEKLM_DAEMON"
The MOSEKLM license manager daemon is not up and running.

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

"MSK_RES_ERR_LICENSE_CANNOT_ALLOCATE"
The license system cannot allocate the memory required.

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

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

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

"MSK_RES_ERR_LICENSE_NO_SERVER_SUPPORT"
The license server does not support the requested feature. Possible reasons for this error include:

- The feature has expired.
- The feature's start date is later than today's date.
- The version requested is higher than feature's the highest supported version.
- A corrupted license file.

Try restarting the license and inspect the license server debug file, usually called `lmgrd.log`.

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

"MSK_RES_ERR_OPEN_DL"
: A dynamic link library could not be opened.

"MSK_RES_ERR_OLDER_DLL"
: The dynamic link library is older than the specified version.

"MSK_RES_ERR_NEWER_DLL"
: The dynamic link library is newer than the specified version.

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

"MSK_RES_ERR_THREAD_MUTEX_INIT"
: Could not initialize a mutex.

"MSK_RES_ERR_THREAD_MUTEX_LOCK"
: Could not lock a mutex.

"MSK_RES_ERR_THREAD_MUTEX_UNLOCK"
: Could not unlock a mutex.

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

"MSK_RES_ERR_THREAD_COND_INIT"
: Could not initialize a condition.

"MSK_RES_ERR_UNKNOWN"
: Unknown error.

"MSK_RES_ERR_SPACE"
: Out of space.

"MSK_RES_ERR_FILE_OPEN"
: Error while opening a file.

"MSK_RES_ERR_FILE_READ"
: File read error.

"MSK_RES_ERR_FILE_WRITE"
: File write error.

"MSK_RES_ERR_DATA_FILE_EXT"
: The data file format cannot be determined from the file name.

"MSK_RES_ERR_INVALID_FILE_NAME"
: An invalid file name has been specified.

"MSK_RES_ERR_INVALID_SOL_FILE_NAME"
: An invalid file name has been specified.

"MSK_RES_ERR_END_OF_FILE"
: End of file reached.

"MSK_RES_ERR_NULL_ENV"
: env is a NULL pointer.

"MSK_RES_ERR_NULL_TASK"
: task is a NULL pointer.

"MSK_RES_ERR_INVALID_STREAM"
: An invalid stream is referenced.

"MSK_RES_ERR_NO_INIT_ENV"
env is not initialized.

"MSK_RES_ERR_INVALID_TASK"
The task is invalid.

"MSK_RES_ERR_NULL_POINTER"
An argument to a function is unexpectedly a NULL pointer.

"MSK_RES_ERR_LIVING_TASKS"
All tasks associated with an enviroment must be deleted before the environment is deleted. There are still some undeleted tasks.

"MSK_RES_ERR_BLANK_NAME"
An all blank name has been specified.

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

"MSK_RES_ERR_INVALID_OBJ_NAME"
An invalid objective name is specified.

"MSK_RES_ERR_INVALID_CON_NAME"
An invalid constraint name is used.

"MSK_RES_ERR_INVALID_VAR_NAME"
An invalid variable name is used.

"MSK_RES_ERR_INVALID_CONE_NAME"
An invalid cone name is used.

"MSK_RES_ERR_INVALID_BARVAR_NAME"
An invalid symmetric matrix variable name is used.

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

"MSK_RES_ERR_SPACE_NO_INFO"
No available information about the space usage.

"MSK_RES_ERR_READ_FORMAT"
The specified format cannot be read.

"MSK_RES_ERR_MPS_FILE"
An error occurred while reading an MPS file.

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

"MSK_RES_ERR_MPS_INV_MARKER"
An invalid marker has been specified in the MPS file.

"MSK_RES_ERR_MPS_NULL_CON_NAME"
An empty constraint name is used in an MPS file.

"MSK_RES_ERR_MPS_NULL_VAR_NAME"
An empty variable name is used in an MPS file.

"MSK_RES_ERR_MPS_UNDEF_CON_NAME"
An undefined constraint name occurred in an MPS file.

"MSK_RES_ERR_MPS_UNDEF_VAR_NAME"
An undefined variable name occurred in an MPS file.

"MSK_RES_ERR_MPS_INV_CON_KEY"
An invalid constraint key occurred in an MPS file.

"MSK_RES_ERR_MPS_INV_BOUND_KEY"
An invalid bound key occurred in an MPS file.

"MSK_RES_ERR_MPS_INV_SEC_NAME"
An invalid section name occurred in an MPS file.

"MSK_RES_ERR_MPS_NO_OBJECTIVE"
No objective is defined in an MPS file.

"MSK_RES_ERR_MPS_SPLITTED_VAR"
All elements in a column of the $A$ matrix must be specified consecutively. Hence, it is illegal to specify non-zero elements in $A$ for variable 1, then for variable 2 and then variable 1 again.

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

"MSK_RES_ERR_MPS_MUL_QSEC"
Multiple QSECTIONs are specified for a constraint in the MPS data file.

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

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

"MSK_RES_ERR_MPS_MUL_CSEC"
Multiple CSECTIONs are given the same name.

"MSK_RES_ERR_MPS_CONE_TYPE"
Invalid cone type specified in a CSECTION.

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

"MSK_RES_ERR_MPS_CONE_REPEAT"
A variable is repeated within the CSECTION.

"MSK_RES_ERR_MPS_NON_SYMMETRIC_Q"
A non symmetric matrice has been speciefied.

"MSK_RES_ERR_MPS_DUPLICATE_Q_ELEMENT"
Duplicate elements is specfied in a $Q$ matrix.

"MSK_RES_ERR_MPS_INVALID_OBJSENSE"
An invalid objective sense is specified.

"MSK_RES_ERR_MPS_TAB_IN_FIELD2"
A tab char occurred in field 2.

"MSK_RES_ERR_MPS_TAB_IN_FIELD3"
A tab char occurred in field 3.

"MSK_RES_ERR_MPS_TAB_IN_FIELD5"
A tab char occurred in field 5.

"MSK_RES_ERR_MPS_INVALID_OBJ_NAME"
An invalid objective name is specified.

"MSK_RES_ERR_LP_INCOMPATIBLE"
The problem cannot be written to an LP formatted file.

"MSK_RES_ERR_LP_EMPTY"
The problem cannot be written to an LP formatted file.

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

"MSK_RES_ERR_WRITE_MPS_INVALID_NAME"
An invalid name is created while writing an MPS file. Usually this will make the MPS file unreadable.

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

"MSK_RES_ERR_LP_FREE_CONSTRAINT"
Free constraints cannot be written in LP file format.

"MSK_RES_ERR_WRITE_OPF_INVALID_VAR_NAME"
Empty variable names cannot be written to OPF files.

"MSK_RES_ERR_LP_FILE_FORMAT"
Syntax error in an LP file.

"MSK_RES_ERR_WRITE_LP_FORMAT"
Problem cannot be written as an LP file.

"MSK_RES_ERR_READ_LP_MISSING_END_TAG"
Syntax error in LP file. Possibly missing End tag.

"MSK_RES_ERR_LP_FORMAT"
Syntax error in an LP file.

"MSK_RES_ERR_WRITE_LP_NON_UNIQUE_NAME"
An auto-generated name is not unique.

"MSK_RES_ERR_READ_LP_NONEXISTING_NAME"
A variable never occurred in objective or constraints.

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

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

"MSK_RES_ERR_WRITING_FILE"
An error occurred while writing file

"MSK_RES_ERR_OPF_FORMAT"
Syntax error in an OPF file

"MSK_RES_ERR_OPF_NEW_VARIABLE"
Introducing new variables is now allowed. When a `[variables]` section is present, it is not allowed to introduce new variables later in the problem.

"MSK_RES_ERR_INVALID_NAME_IN_SOL_FILE"
An invalid name occurred in a solution file.

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

"MSK_RES_ERR_OPF_PREMATURE_EOF"
Premature end of file in an OPF file.

"MSK_RES_ERR_JSON_SYNTAX"
Syntax error in an JSON data

"MSK_RES_ERR_JSON_STRING"
Error in JSON string.

"MSK_RES_ERR_JSON_NUMBER_OVERFLOW"
Invalid number entry - wrong type or value overflow.

"MSK_RES_ERR_JSON_FORMAT"
Error in an JSON Task file

"MSK_RES_ERR_JSON_DATA"
Inconsistent data in JSON Task file

"MSK_RES_ERR_JSON_MISSING_DATA"
Missing data section in JSON task file.

"MSK_RES_ERR_ARGUMENT_LENNEQ"
Incorrect length of arguments.

"MSK_RES_ERR_ARGUMENT_TYPE"
Incorrect argument type.

"MSK_RES_ERR_NR_ARGUMENTS"
Incorrect number of function arguments.

"MSK_RES_ERR_IN_ARGUMENT"
A function argument is incorrect.

"MSK_RES_ERR_ARGUMENT_DIMENSION"
A function argument is of incorrect dimension.

"MSK_RES_ERR_INDEX_IS_TOO_SMALL"
An index in an argument is too small.

"MSK_RES_ERR_INDEX_IS_TOO_LARGE"
An index in an argument is too large.

"MSK_RES_ERR_PARAM_NAME"
The parameter name is not correct.

"MSK_RES_ERR_PARAM_NAME_DOU"
The parameter name is not correct for a double parameter.

"MSK_RES_ERR_PARAM_NAME_INT"
The parameter name is not correct for an integer parameter.

"MSK_RES_ERR_PARAM_NAME_STR"
The parameter name is not correct for a string parameter.

"MSK_RES_ERR_PARAM_INDEX"
Parameter index is out of range.

"MSK_RES_ERR_PARAM_IS_TOO_LARGE"
The parameter value is too large.

"MSK_RES_ERR_PARAM_IS_TOO_SMALL"
The parameter value is too small.

"MSK_RES_ERR_PARAM_VALUE_STR"
The parameter value string is incorrect.

"MSK_RES_ERR_PARAM_TYPE"
The parameter type is invalid.

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

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

"MSK_RES_ERR_INDEX_ARR_IS_TOO_SMALL"
An index in an array argument is too small.

"MSK_RES_ERR_INDEX_ARR_IS_TOO_LARGE"
An index in an array argument is too large.

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

"MSK_RES_ERR_ARG_IS_TOO_SMALL"
The value of a argument is too small.

"MSK_RES_ERR_ARG_IS_TOO_LARGE"
The value of a argument is too small.

"MSK_RES_ERR_INVALID_WHICHSOL"
: `whichsol` is invalid.

"MSK_RES_ERR_INF_DOU_NAME"
: A double information name is invalid.

"MSK_RES_ERR_INF_INT_NAME"
: An integer information name is invalid.

"MSK_RES_ERR_INF_TYPE"
: The information type is invalid.

"MSK_RES_ERR_INF_LINT_NAME"
: A long integer information name is invalid.

"MSK_RES_ERR_INDEX"
: An index is out of range.

"MSK_RES_ERR_WHICHSOL"
: The solution defined by `whichsol` does not exists.

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

"MSK_RES_ERR_WHICHITEM_NOT_ALLOWED"
: *whichitem* is unacceptable.

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

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

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

"MSK_RES_ERR_MAXNUMQNZ"
: The maximum number of non-zeros specified for the $Q$ matrices is smaller than the number of non-zeros in the current $Q$ matrices.

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

"MSK_RES_ERR_INVALID_IDX"
: A specified index is invalid.

"MSK_RES_ERR_INVALID_MAX_NUM"
: A specified index is invalid.

"MSK_RES_ERR_NUMCONLIM"
: Maximum number of constraints limit is exceeded.

"MSK_RES_ERR_NUMVARLIM"
: Maximum number of variables limit is exceeded.

"MSK_RES_ERR_TOO_SMALL_MAXNUMANZ"
: The maximum number of non-zeros specified for $A$ is smaller than the number of non-zeros in the current $A$.

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

"MSK_RES_ERR_MUL_A_ELEMENT"
: An element in $A$ is defined multiple times.

"MSK_RES_ERR_INV_BK"
Invalid bound key.

"MSK_RES_ERR_INV_BKC"
Invalid bound key is specified for a constraint.

"MSK_RES_ERR_INV_BKX"
An invalid bound key is specified for a variable.

"MSK_RES_ERR_INV_VAR_TYPE"
An invalid variable type is specified for a variable.

"MSK_RES_ERR_SOLVER_PROBTYPE"
Problem type does not match the chosen optimizer.

"MSK_RES_ERR_OBJECTIVE_RANGE"
Empty objective range.

"MSK_RES_ERR_FIRST"
Invalid `first`.

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

"MSK_RES_ERR_NEGATIVE_SURPLUS"
Negative surplus.

"MSK_RES_ERR_NEGATIVE_APPEND"
Cannot append a negative number.

"MSK_RES_ERR_UNDEF_SOLUTION"
**MOSEK** has the following solution types:

- an interior-point solution,
- an basic solution,
- and an integer solution.

Each optimizer may set one or more of these solutions; e.g by default a successful optimization with the interior-point optimizer defines the interior-point solution, and, for linear problems, also the basic solution. This error occurs when asking for a solution or for information about a solution that is not defined.

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

"MSK_RES_ERR_INV_SKC"
Invalid value in `skc`.

"MSK_RES_ERR_INV_SKX"
Invalid value in `skx`.

"MSK_RES_ERR_INV_SKN"
Invalid value in `skn`.

"MSK_RES_ERR_INV_SK_STR"
Invalid status key string encountered.

"MSK_RES_ERR_INV_SK"
Invalid status key code.

"MSK_RES_ERR_INV_CONE_TYPE_STR"
Invalid cone type string encountered.

"MSK_RES_ERR_INV_CONE_TYPE"
Invalid cone type code is encountered.

"MSK_RES_ERR_INVALID_SURPLUS"
Invalid surplus.

"MSK_RES_ERR_INV_NAME_ITEM"
An invalid name item code is used.

"MSK_RES_ERR_PRO_ITEM"
An invalid problem is used.

"MSK_RES_ERR_INVALID_FORMAT_TYPE"
Invalid format type.

"MSK_RES_ERR_FIRSTI"
Invalid `firsti`.

"MSK_RES_ERR_LASTI"
Invalid `lasti`.

"MSK_RES_ERR_FIRSTJ"
Invalid `firstj`.

"MSK_RES_ERR_LASTJ"
Invalid `lastj`.

"MSK_RES_ERR_MAX_LEN_IS_TOO_SMALL"
An maximum length that is too small has been specified.

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

"MSK_RES_ERR_NONCONVEX"
The optimization problem is nonconvex.

"MSK_RES_ERR_NONLINEAR_RANGED"
Nonlinear constraints with finite lower and upper bound always define a nonconvex feasible set.

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

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

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

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

"MSK_RES_ERR_ARGUMENT_PERM_ARRAY"
An invalid permutation array is specified.

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

"MSK_RES_ERR_CONE_SIZE"
A cone with too few members is specified.

"MSK_RES_ERR_CONE_OVERLAP"
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"
A variable is included multiple times in the cone.

"MSK_RES_ERR_MAXNUMCONE"
The value specified for maxnumcone is too small.

"MSK_RES_ERR_CONE_TYPE"
Invalid cone type specified.

"MSK_RES_ERR_CONE_TYPE_STR"
Invalid cone type specified.

"MSK_RES_ERR_CONE_OVERLAP_APPEND"
The cone to be appended has one variable which is already member of another cone.

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

"MSK_RES_ERR_SOL_FILE_INVALID_NUMBER"
An invalid number is specified in a solution file.

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

"MSK_RES_ERR_HUGE_AIJ"
A numerically huge value is specified for an $a_{i,j}$ element in $A$. The parameter *MSK_DPAR_DATA_TOL_AIJ_HUGE* controls when an $a_{i,j}$ is considered huge.

"MSK_RES_ERR_DUPLICATE_AIJ"
An element in the A matrix is specified twice.

"MSK_RES_ERR_LOWER_BOUND_IS_A_NAN"
The lower bound specified is not a number (nan).

"MSK_RES_ERR_UPPER_BOUND_IS_A_NAN"
The upper bound specified is not a number (nan).

"MSK_RES_ERR_INFINITE_BOUND"
A numerically huge bound value is specified.

"MSK_RES_ERR_INV_QOBJ_SUBI"
Invalid value in qosubi.

"MSK_RES_ERR_INV_QOBJ_SUBJ"
Invalid value in qosubj.

"MSK_RES_ERR_INV_QOBJ_VAL"
Invalid value in qoval.

"MSK_RES_ERR_INV_QCON_SUBK"
Invalid value in qcsubk.

"MSK_RES_ERR_INV_QCON_SUBI"
Invalid value in qcsubi.

"MSK_RES_ERR_INV_QCON_SUBJ"
Invalid value in qcsubj.

"MSK_RES_ERR_INV_QCON_VAL"
Invalid value in qcval.

"MSK_RES_ERR_QCON_SUBI_TOO_SMALL"
Invalid value in qcsubi.

"MSK_RES_ERR_QCON_SUBI_TOO_LARGE"
Invalid value in qcsubi.

"MSK_RES_ERR_QOBJ_UPPER_TRIANGLE"
An element in the upper triangle of $Q^o$ is specified. Only elements in the lower triangle should be specified.

"MSK_RES_ERR_QCON_UPPER_TRIANGLE"
An element in the upper triangle of a $Q^k$ is specified. Only elements in the lower triangle should be specified.

"MSK_RES_ERR_FIXED_BOUND_VALUES"
A fixed constraint/variable has been specified using the bound keys but the numerical value of the lower and upper bound is different.

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

"MSK_RES_ERR_USER_FUNC_RET"
An user function reported an error.

"MSK_RES_ERR_USER_FUNC_RET_DATA"
An user function returned invalid data.

"MSK_RES_ERR_USER_NLO_FUNC"
The user-defined nonlinear function reported an error.

"MSK_RES_ERR_USER_NLO_EVAL"
The user-defined nonlinear function reported an error.

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

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

"MSK_RES_ERR_INVALID_OBJECTIVE_SENSE"
An invalid objective sense is specified.

"MSK_RES_ERR_UNDEFINED_OBJECTIVE_SENSE"
The objective sense has not been specified before the optimization.

"MSK_RES_ERR_Y_IS_UNDEFINED"
The solution item $y$ is undefined.

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

"MSK_RES_ERR_NAN_IN_BLC"
$l^c$ contains an invalid floating point value, i.e. a NaN.

"MSK_RES_ERR_NAN_IN_BUC"
$u^c$ contains an invalid floating point value, i.e. a NaN.

"MSK_RES_ERR_NAN_IN_C"
$c$ contains an invalid floating point value, i.e. a NaN.

"MSK_RES_ERR_NAN_IN_BLX"
$l^x$ contains an invalid floating point value, i.e. a NaN.

"MSK_RES_ERR_NAN_IN_BUX"
$u^x$ contains an invalid floating point value, i.e. a NaN.

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

"MSK_RES_ERR_SYM_MAT_INVALID"
A symmetric matrix contains an invalid floating point value, i.e. a NaN or an infinite value.

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

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

"MSK_RES_ERR_INV_OPTIMIZER"
An invalid optimizer has been chosen for the problem. This means that the simplex or the conic optimizer is chosen to optimize a nonlinear problem.

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

"MSK_RES_ERR_NO_OPTIMIZER_VAR_TYPE"
No optimizer is available for this class of optimization problems.

"MSK_RES_ERR_FINAL_SOLUTION"
An error occurred during the solution finalization.

"MSK_RES_ERR_POSTSOLVE"
An error occurred during the postsolve. Please contact **MOSEK** support.

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

"MSK_RES_ERR_NO_BASIS_SOL"
No basic solution is defined.

"MSK_RES_ERR_BASIS_FACTOR"
The factorization of the basis is invalid.

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

"MSK_RES_ERR_FACTOR"
An error occurred while factorizing a matrix.

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

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

"MSK_RES_ERR_FEASREPAIR_INCONSISTENT_BOUND"
The upper bound is less than the lower bound for a variable or a constraint. Please correct this before running the feasibility repair.

"MSK_RES_ERR_REPAIR_INVALID_PROBLEM"
The feasibility repair does not support the specified problem type.

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

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

"MSK_RES_ERR_NAME_IS_NULL"
The name buffer is a NULL pointer.

"MSK_RES_ERR_INVALID_COMPRESSION"
Invalid compression type.

"MSK_RES_ERR_INVALID_IOMODE"
Invalid io mode.

"MSK_RES_ERR_NO_PRIMAL_INFEAS_CER"
A certificate of primal infeasibility is not available.

"MSK_RES_ERR_NO_DUAL_INFEAS_CER"
A certificate of infeasibility is not available.

"MSK_RES_ERR_NO_SOLUTION_IN_CALLBACK"
The required solution is not available.

"MSK_RES_ERR_INV_MARKI"
Invalid value in marki.

"MSK_RES_ERR_INV_MARKJ"
Invalid value in markj.

"MSK_RES_ERR_INV_NUMI"
Invalid numi.

"MSK_RES_ERR_INV_NUMJ"
Invalid numj.

"MSK_RES_ERR_CANNOT_CLONE_NL"
A task with a nonlinear function callback cannot be cloned.

"MSK_RES_ERR_CANNOT_HANDLE_NL"
A function cannot handle a task with nonlinear function callbacks.

"MSK_RES_ERR_INVALID_ACCMODE"
An invalid access mode is specified.

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

"MSK_RES_ERR_TASK_INVALID"
The Task file is invalid.

"MSK_RES_ERR_TASK_WRITE"
Failed to write the task file.

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

"MSK_RES_ERR_INVALID_UTF8"
An invalid UTF8 string is encountered.

"MSK_RES_ERR_INVALID_WCHAR"
An invalid `wchar` string is encountered.

"MSK_RES_ERR_NO_DUAL_FOR_ITG_SOL"
No dual information is available for the integer solution.

"MSK_RES_ERR_NO_SNX_FOR_BAS_SOL"
$s_n^x$ is not available for the basis solution.

"MSK_RES_ERR_INTERNAL"
An internal error occurred. Please report this problem.

"MSK_RES_ERR_API_ARRAY_TOO_SMALL"
An input array was too short.

"MSK_RES_ERR_API_CB_CONNECT"
Failed to connect a callback object.

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

"MSK_RES_ERR_API_INTERNAL"
An internal fatal error occurred in an interface function.

"MSK_RES_ERR_SEN_FORMAT"
Syntax error in sensitivity analysis file.

"MSK_RES_ERR_SEN_UNDEF_NAME"
An undefined name was encountered in the sensitivity analysis file.

"MSK_RES_ERR_SEN_INDEX_RANGE"
Index out of range in the sensitivity analysis file.

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

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

"MSK_RES_ERR_SEN_INDEX_INVALID"
Invalid range given in the sensitivity file.

"MSK_RES_ERR_SEN_INVALID_REGEXP"
Syntax error in regexp or regexp longer than 1024.

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

"MSK_RES_ERR_SEN_NUMERICAL"
Numerical difficulties encountered performing the sensitivity analysis.

"MSK_RES_ERR_SEN_UNHANDLED_PROBLEM_TYPE"
Sensitivity analysis cannot be performed for the specified problem. Sensitivity analysis is only possible for linear problems.

"MSK_RES_ERR_UNB_STEP_SIZE"
A step size in an optimizer was unexpectedly unbounded. For instance, if the step-size becomes unbounded in phase 1 of the simplex algorithm then an error occurs. Normally this will happen only if the problem is badly formulated. Please contact **MOSEK** support if this error occurs.

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

"MSK_RES_ERR_AD_INVALID_CODELIST"
The code list data was invalid.

"MSK_RES_ERR_INTERNAL_TEST_FAILED"
An internal unit test function failed.

"MSK_RES_ERR_XML_INVALID_PROBLEM_TYPE"
The problem type is not supported by the XML format.

"MSK_RES_ERR_INVALID_AMPL_STUB"
Invalid AMPL stub.

"MSK_RES_ERR_INT64_TO_INT32_CAST"
An 32 bit integer could not cast to a 64 bit integer.

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

"MSK_RES_ERR_INFEAS_UNDEFINED"
The requested value is not defined for this solution type.

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

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

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

"MSK_RES_ERR_SYM_MAT_INVALID_ROW_INDEX"
A row index specified for sparse symmetric matrix is invalid.

"MSK_RES_ERR_SYM_MAT_INVALID_COL_INDEX"
A column index specified for sparse symmetric matrix is invalid.

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

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

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

"MSK_RES_ERR_INVALID_SYM_MAT_DIM"
A sparse symmetric matrix of invalid dimension is specified.

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

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

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

"MSK_RES_ERR_DUPLICATE_CONSTRAINT_NAMES"
Two constraint names are identical.

"MSK_RES_ERR_DUPLICATE_VARIABLE_NAMES"
Two variable names are identical.

"MSK_RES_ERR_DUPLICATE_BARVARIABLE_NAMES"
Two barvariable names are identical.

"MSK_RES_ERR_DUPLICATE_CONE_NAMES"
Two cone names are identical.

"MSK_RES_ERR_NON_UNIQUE_ARRAY"
An array does not contain unique elements.

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

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

"MSK_RES_ERR_INVALID_PROBLEM_TYPE"
An invalid problem type.

"MSK_RES_ERR_UNHANDLED_SOLUTION_STATUS"
Unhandled solution status.

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

"MSK_RES_ERR_LAU_SINGULAR_MATRIX"
A matrix is singular.

"MSK_RES_ERR_LAU_NOT_POSITIVE_DEFINITE"
A matrix is not positive definite.

"MSK_RES_ERR_LAU_INVALID_LOWER_TRIANGULAR_MATRIX"
An invalid lower triangular matrix.

"MSK_RES_ERR_LAU_UNKNOWN"
An unknown error.

"MSK_RES_ERR_LAU_ARG_M"
Invalid argument m.

"MSK_RES_ERR_LAU_ARG_N"
Invalid argument n.

"MSK_RES_ERR_LAU_ARG_K"
Invalid argument k.

"MSK_RES_ERR_LAU_ARG_TRANSA"
Invalid argument transa.

"MSK_RES_ERR_LAU_ARG_TRANSB"
Invalid argument transb.

"MSK_RES_ERR_LAU_ARG_UPLO"
Invalid argument uplo.

"MSK_RES_ERR_LAU_ARG_TRANS"
Invalid argument trans.

"MSK_RES_ERR_LAU_INVALID_SPARSE_SYMMETRIC_MATRIX"
An invalid sparse symmetric matrix is specfified. Note only the lower triangular part with no duplicates is specifed.

"MSK_RES_ERR_CBF_PARSE"
An error occurred while parsing an CBF file.

"MSK_RES_ERR_CBF_OBJ_SENSE"
An invalid objective sense is specified.

"MSK_RES_ERR_CBF_NO_VARIABLES"
No variables are specified.

"MSK_RES_ERR_CBF_TOO_MANY_CONSTRAINTS"
Too many constraints specified.

"MSK_RES_ERR_CBF_TOO_MANY_VARIABLES"
Too many variables specified.

"MSK_RES_ERR_CBF_NO_VERSION_SPECIFIED"
No version specified.

"MSK_RES_ERR_CBF_SYNTAX"
Invalid syntax.

"MSK_RES_ERR_CBF_DUPLICATE_OBJ"
Duplicate OBJ keyword.

"MSK_RES_ERR_CBF_DUPLICATE_CON"
Duplicate CON keyword.

"MSK_RES_ERR_CBF_DUPLICATE_VAR"
Duplicate VAR keyword.

"MSK_RES_ERR_CBF_DUPLICATE_INT"
Duplicate INT keyword.

"MSK_RES_ERR_CBF_INVALID_VAR_TYPE"
Invalid variable type.

"MSK_RES_ERR_CBF_INVALID_CON_TYPE"
Invalid constraint type.

"MSK_RES_ERR_CBF_INVALID_DOMAIN_DIMENSION"
Invalid domain dimension.

"MSK_RES_ERR_CBF_DUPLICATE_OBJACOORD"
Duplicate index in OBJCOORD.

"MSK_RES_ERR_CBF_DUPLICATE_BCOORD"
Duplicate index in BCOORD.

"MSK_RES_ERR_CBF_DUPLICATE_ACOORD"
Duplicate index in ACOORD.

"MSK_RES_ERR_CBF_TOO_FEW_VARIABLES"
Too few variables defined.

"MSK_RES_ERR_CBF_TOO_FEW_CONSTRAINTS"
Too few constraints defined.

"MSK_RES_ERR_CBF_TOO_FEW_INTS"
Too few ints are specified.

"MSK_RES_ERR_CBF_TOO_MANY_INTS"
Too many ints are specified.

"MSK_RES_ERR_CBF_INVALID_INT_INDEX"
Invalid INT index.

"MSK_RES_ERR_CBF_UNSUPPORTED"
Unsupported feature is present.

"MSK_RES_ERR_CBF_DUPLICATE_PSDVAR"
Duplicate PSDVAR keyword.

"MSK_RES_ERR_CBF_INVALID_PSDVAR_DIMENSION"
Invalid PSDVAR dimmension.

"MSK_RES_ERR_CBF_TOO_FEW_PSDVAR"
Too few variables defined.

"MSK_RES_ERR_MIO_INVALID_ROOT_OPTIMIZER"
An invalid root optimizer was selected for the problem type.

"MSK_RES_ERR_MIO_INVALID_NODE_OPTIMIZER"
An invalid node optimizer was selected for the problem type.

"MSK_RES_ERR_TOCONIC_CONSTR_Q_NOT_PSD"
The matrix defining the quadratric part of constraint is not positive semidefinite.

"MSK_RES_ERR_TOCONIC_CONSTRAINT_FX"
The quadratic constraint is an equality, thus not convex.

"MSK_RES_ERR_TOCONIC_CONSTRAINT_RA"
The quadratic constraint has finite lower and upper bound, and therefore it is not convex.

"MSK_RES_ERR_TOCONIC_CONSTR_NOT_CONIC"
The constraint is not conic representable.

"MSK_RES_ERR_TOCONIC_OBJECTIVE_NOT_PSD"
The matrix defining the quadratric part of the objective function is not positive semidefinite.

"MSK_RES_ERR_SERVER_CONNECT"
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"
Unexpected message or data from solver server.

"MSK_RES_ERR_SERVER_STATUS"
Server returned non-ok HTTP status code

"MSK_RES_ERR_SERVER_TOKEN"
The job ID specified is incorrect or invalid

## 17.6 Enumerations

`language`
Language selection constants

`"MSK_LANG_ENG"`
English language selection

`"MSK_LANG_DAN"`
Danish language selection

`accmode`
Constraint or variable access modes. All functions using this enum are deprecated. Use separate functions for rows/columns instead.

`"MSK_ACC_VAR"`
Access data by columns (variable oriented)

`"MSK_ACC_CON"`
Access data by rows (constraint oriented)

`basindtype`
Basis identification

`"MSK_BI_NEVER"`
Never do basis identification.

`"MSK_BI_ALWAYS"`
Basis identification is always performed even if the interior-point optimizer terminates abnormally.

`"MSK_BI_NO_ERROR"`
Basis identification is performed if the interior-point optimizer terminates without an error.

`"MSK_BI_IF_FEASIBLE"`
Basis identification is not performed if the interior-point optimizer terminates with a problem status saying that the problem is primal or dual infeasible.

`"MSK_BI_RESERVERED"`
Not currently in use.

`boundkey`
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.

`mark`
Mark

`"MSK_MARK_LO"`
The lower bound is selected for sensitivity analysis.

`"MSK_MARK_UP"`
The upper bound is selected for sensitivity analysis.

`simdegen`
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.

`transpose`
Transposed matrix.

`"MSK_TRANSPOSE_NO"`
No transpose is applied.

`"MSK_TRANSPOSE_YES"`
A transpose is applied.

`uplo`
Triangular part of a symmetric matrix.

`"MSK_UPLO_LO"`
Lower part.

`"MSK_UPLO_UP"`
Upper part

`simreform`
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.

`simdupvec`
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.

`simhotstart`
Hot-start type employed by the simplex optimizer

`"MSK_SIM_HOTSTART_NONE"`
The simplex optimizer performs a coldstart.

"MSK_SIM_HOTSTART_FREE"
: The simplex optimize chooses the hot-start type.

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

intpnthotstart
: 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.

callbackcode
: 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_FULL_CONVEXITY_CHECK"
: Begin full convexity check.

"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_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_FULL_CONVEXITY_CHECK"
End full convexity check.

"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 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_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_FULL_CONVEXITY_CHECK"
The callback function is called at an intermediate stage of the full convexity check.

"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_WRITE_OPF"
: The callback function is called from the OPF writer.

checkconvexitytype
: 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.

**compresstype**
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.

**conetype**
Cone types

"MSK_CT_QUAD"
: The cone is a quadratic cone.

"MSK_CT_RQUAD"
: The cone is a rotated quadratic cone.

**nametype**
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.

**symmattype**
Cone types

"MSK_SYMMAT_TYPE_SPARSE"
: Sparse symmetric matrix.

**dataformat**
Data format types

"MSK_DATA_FORMAT_EXTENSION"
: The file extension is used to determine the data file format.

"MSK_DATA_FORMAT_MPS"
: The data file is MPS formatted.

"MSK_DATA_FORMAT_LP"
: The data file is LP formatted.

"MSK_DATA_FORMAT_OP"
: The data file is an optimization problem formatted file.

"MSK_DATA_FORMAT_XML"
: The data file is an XML formatted file.

"MSK_DATA_FORMAT_FREE_MPS"
: The data a free MPS formatted file.

"MSK_DATA_FORMAT_TASK"
Generic task dump file.

"MSK_DATA_FORMAT_CB"
Conic benchmark format,

"MSK_DATA_FORMAT_JSON_TASK"
JSON based task format.

**dinfitem**
Double information items

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

"MSK_DINF_BI_CLEAN_PRIMAL_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_CLIQUE_SEPARATION_TIME"
Seperation time for clique cuts.

"MSK_DINF_MIO_CMIR_SEPARATION_TIME"
Seperation time for CMIR cuts.

"MSK_DINF_MIO_CONSTRUCT_SOLUTION_OBJ"
If **MOSEK** has successfully constructed an integer feasible solution, then this item contains the optimal objective value corresponding to the feasible solution.

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

"MSK_DINF_MIO_GMI_SEPARATION_TIME"
Seperation time for GMI cuts.

"MSK_DINF_MIO_HEURISTIC_TIME"
Total time spent in the optimizer.

"MSK_DINF_MIO_IMPLIED_BOUND_TIME"
Seperation time for implied bound cuts.

"MSK_DINF_MIO_KNAPSACK_COVER_SEPARATION_TIME"
Seperation time for knapsack cover.

"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 $\delta$ is given by the parameter *MSK_DPAR_MIO_REL_GAP_CONST*. Otherwise it has the value $-1.0$.

"MSK_DINF_MIO_OPTIMIZER_TIME"
Total time spent in the optimizer.

"MSK_DINF_MIO_PROBING_TIME"
Total time for probing.

"MSK_DINF_MIO_ROOT_CUTGEN_TIME"
Total time for cut generation.

"MSK_DINF_MIO_ROOT_OPTIMIZER_TIME"
Time spent in the optimizer while solving the root relaxation.

"MSK_DINF_MIO_ROOT_PRESOLVE_TIME"
Time spent in while presolving the root relaxation.

"MSK_DINF_MIO_TIME"
Time spent in the mixed-integer optimizer.

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

"MSK_DINF_OPTIMIZER_TIME"
Total time spent in the optimizer since it was invoked.

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

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

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

"MSK_DINF_PRIMAL_REPAIR_PENALTY_OBJ"
The optimal objective value of the penalty function.

"MSK_DINF_QCQO_REFORMULATE_MAX_PERTURBATION"
Maximum absolute diagonal perturbation occuring 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_RD_TIME"
Time spent reading the data file.

"MSK_DINF_SIM_DUAL_TIME"
Time spent in the dual simplex optimizer since invoking it.

"MSK_DINF_SIM_FEAS"
Feasibility measure reported by the simplex optimizer.

"MSK_DINF_SIM_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.

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

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

"MSK_DINF_SOL_BAS_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.

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

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

"MSK_DINF_SOL_ITG_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.

"MSK_DINF_SOL_ITG_PVIOLBARVAR"
Maximal primal bound violation for $\overline{X}$ in the integer solution.

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

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

"MSK_DINF_SOL_ITG_PVIOLITG"
Maximal violation for the integer constraints in the integer solution.

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

"MSK_DINF_SOL_ITR_DUAL_OBJ"
Dual objective value of the interior-point solution.

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

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

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

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

"MSK_DINF_SOL_ITR_NRM_BARS"
Infinity norm of $\overline{S}$ in the interior-point solution.

"MSK_DINF_SOL_ITR_NRM_BARX"
Infinity norm of $\overline{X}$ 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.

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

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

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

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

"MSK_DINF_TO_CONIC_TIME"
Time spent in the last to conic reformulation.

feature
License feature

"MSK_FEATURE_PTS"
Base system.

"MSK_FEATURE_PTON"
Nonlinear extension.

liinfitem
Long integer information items.

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

"MSK_LIINF_BI_CLEAN_DUAL_ITER"
Number of dual clean iterations performed in the basis identification.

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

"MSK_LIINF_BI_CLEAN_PRIMAL_ITER"
Number of primal clean iterations performed in the basis identification.

"MSK_LIINF_BI_DUAL_ITER"
Number of dual pivots performed in the basis identification.

"MSK_LIINF_BI_PRIMAL_ITER"
Number of primal pivots performed in the basis identification.

"MSK_LIINF_INTPNT_FACTOR_NUM_NZ"
Number of non-zeros in factorization.

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

"MSK_LIINF_MIO_PRESOLVED_ANZ"
Number of non-zero entries in the constraint matrix of presolved problem.

"MSK_LIINF_MIO_SIM_MAXITER_SETBACKS"
Number of times the the simplex optimizer has hit the maximum iteration limit when re-optimizing.

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

"MSK_LIINF_RD_NUMANZ"
Number of non-zeros in A that is read.

"MSK_LIINF_RD_NUMQNZ"
Number of Q non-zeros.

iinfitem
Integer information items.

"MSK_IINF_ANA_PRO_NUM_CON"
Number of constraints in the problem.

"MSK_IINF_ANA_PRO_NUM_CON_EQ"
Number of equality constraints.

"MSK_IINF_ANA_PRO_NUM_CON_FR"
Number of unbounded constraints.

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

"MSK_IINF_ANA_PRO_NUM_CON_RA"
Number of constraints with finite lower and upper bounds.

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

"MSK_IINF_ANA_PRO_NUM_VAR"
Number of variables in the problem.

"MSK_IINF_ANA_PRO_NUM_VAR_BIN"
Number of binary (0-1) variables.

"MSK_IINF_ANA_PRO_NUM_VAR_CONT"
Number of continuous variables.

"MSK_IINF_ANA_PRO_NUM_VAR_EQ"
Number of fixed variables.

"MSK_IINF_ANA_PRO_NUM_VAR_FR"
Number of free variables.

"MSK_IINF_ANA_PRO_NUM_VAR_INT"
Number of general integer variables.

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

"MSK_IINF_ANA_PRO_NUM_VAR_RA"
Number of variables with finite lower and upper bounds.

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

"MSK_IINF_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_CLIQUE_TABLE_SIZE"
Size of the clique table.

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

"MSK_IINF_MIO_CONSTRUCT_SOLUTION"
If this item has the value 0, then **MOSEK** did not try to construct an initial integer feasible solution. If the item has a positive value, then **MOSEK** successfully constructed an initial integer feasible solution.

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

"MSK_IINF_MIO_NEAR_ABSGAP_SATISFIED"
Non-zero if absolute gap is within relaxed tolerances.

"MSK_IINF_MIO_NEAR_RELGAP_SATISFIED"
Non-zero if relative gap is within relaxed tolerances.

"MSK_IINF_MIO_NODE_DEPTH"
Depth of the last node solved.

"MSK_IINF_MIO_NUM_ACTIVE_NODES"
Number of active branch 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 has been found.

"MSK_IINF_MIO_NUM_KNAPSACK_COVER_CUTS"
Number of clique 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_NUMCON"
Number of constraints in the problem solved by the mixed-integer optimizer.

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

"MSK_IINF_MIO_NUMVAR"
Number of variables in the problem solved 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_PRESOLVED_NUMBIN"
Number of binary variables in the problem solved be the mixed-integer optimizer.

"MSK_IINF_MIO_PRESOLVED_NUMCON"
Number of constraints in the presolved problem.

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

"MSK_IINF_MIO_PRESOLVED_NUMINT"
Number of integer variables in the presolved problem.

"MSK_IINF_MIO_PRESOLVED_NUMVAR"
Number of variables in the presolved problem.

"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_RD_NUMBARVAR"
Number of variables read.

"MSK_IINF_RD_NUMCON"
Number of constraints read.

"MSK_IINF_RD_NUMCONE"
Number of conic constraints read.

"MSK_IINF_RD_NUMINTVAR"
Number of integer-constrained variables read.

"MSK_IINF_RD_NUMQ"
Number of nonempty Q matrices read.

"MSK_IINF_RD_NUMVAR"
Number of variables read.

"MSK_IINF_RD_PROTYPE"
Problem type.

"MSK_IINF_SIM_DUAL_DEG_ITER"
The number of dual degenerate iterations.

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

"MSK_IINF_SIM_DUAL_HOTSTART_LU"
If 1 then a valid basis factorization of full rank was located and used by the dual simplex algorithm.

"MSK_IINF_SIM_DUAL_INF_ITER"
The number of iterations taken with dual infeasibility.

"MSK_IINF_SIM_DUAL_ITER"
Number of dual simplex iterations during the last optimization.

"MSK_IINF_SIM_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 indicates that memory fragmentation may occur.

inftype
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.

iomode
Input/output modes

"MSK_IOMODE_READ"
The file is read-only.

"MSK_IOMODE_WRITE"
The file is write-only. If the file exists then it is truncated when it is opened. Otherwise it is created when it is opened.

"MSK_IOMODE_READWRITE"
The file is to read and written.

branchdir
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.

miocontsoltype
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.

miomode
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.

mionodeseltype
Mixed-integer node selection types

"MSK_MIO_NODE_SELECTION_FREE"
The optimizer decides the node selection strategy.

"MSK_MIO_NODE_SELECTION_FIRST"
The optimizer employs a depth first node selection strategy.

"MSK_MIO_NODE_SELECTION_BEST"
The optimizer employs a best bound node selection strategy.

"MSK_MIO_NODE_SELECTION_WORST"
The optimizer employs a worst bound node selection strategy.

"MSK_MIO_NODE_SELECTION_HYBRID"
The optimizer employs a hybrid strategy.

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

mpsformat
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.

objsense
Objective sense types

"MSK_OBJECTIVE_SENSE_MINIMIZE"
The problem should be minimized.

"MSK_OBJECTIVE_SENSE_MAXIMIZE"
The problem should be maximized.

onoffkey
On/off

"MSK_ON"
Switch the option on.

"MSK_OFF"
Switch the option off.

optimizertype
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.

orderingtype
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.

presolvemode
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.

parametertype
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.

problemitem
Problem data items

"MSK_PI_VAR"
Item is a variable.

"MSK_PI_CON"
: Item is a constraint.

"MSK_PI_CONE"
: Item is a cone.

problemtype
: Problem types

"MSK_PROBTYPE_LO"
: The problem is a linear optimization problem.

"MSK_PROBTYPE_QO"
: The problem is a quadratic optimization problem.

"MSK_PROBTYPE_QCQO"
: The problem is a quadratically constrained optimization problem.

"MSK_PROBTYPE_GECO"
: General convex optimization.

"MSK_PROBTYPE_CONIC"
: A conic optimization.

"MSK_PROBTYPE_MIXED"
: General nonlinear constraints and conic constraints. This combination can not be solved by **MOSEK**.

prosta
: 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_NEAR_PRIM_AND_DUAL_FEAS"
: The problem is at least nearly primal and dual feasible.

"MSK_PRO_STA_NEAR_PRIM_FEAS"
: The problem is at least nearly primal feasible.

"MSK_PRO_STA_NEAR_DUAL_FEAS"
: The problem is at least nearly dual feasible.

"MSK_PRO_STA_PRIM_INFEAS"
: 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.

`xmlwriteroutputtype`
XML writer output mode

`"MSK_WRITE_XML_MODE_ROW"`
Write in row order.

`"MSK_WRITE_XML_MODE_COL"`
Write in column order.

`rescodetype`
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.

`scalingtype`
Scaling type

`"MSK_SCALING_FREE"`
The optimizer chooses the scaling heuristic.

`"MSK_SCALING_NONE"`
No scaling is performed.

`"MSK_SCALING_MODERATE"`
A conservative scaling is performed.

`"MSK_SCALING_AGGRESSIVE"`
A very aggressive scaling is performed.

`scalingmethod`
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.

`sensitivitytype`
Sensitivity types

`"MSK_SENSITIVITY_TYPE_BASIS"`
Basis sensitivity analysis is performed.

`"MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION"`
Optimal partition sensitivity analysis is performed.

`simseltype`
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.

solitem
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.

solsta
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_NEAR_OPTIMAL"
The solution is nearly optimal.

"MSK_SOL_STA_NEAR_PRIM_FEAS"
The solution is nearly primal feasible.

"MSK_SOL_STA_NEAR_DUAL_FEAS"
The solution is nearly dual feasible.

"MSK_SOL_STA_NEAR_PRIM_AND_DUAL_FEAS"
The solution is nearly both primal and dual feasible.

"MSK_SOL_STA_PRIM_INFEAS_CER"
The solution is a certificate of primal infeasibility.

"MSK_SOL_STA_DUAL_INFEAS_CER"
The solution is a certificate of dual infeasibility.

"MSK_SOL_STA_NEAR_PRIM_INFEAS_CER"
The solution is almost a certificate of primal infeasibility.

"MSK_SOL_STA_NEAR_DUAL_INFEAS_CER"
The solution is almost a certificate of dual infeasibility.

"MSK_SOL_STA_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.

"MSK_SOL_STA_NEAR_INTEGER_OPTIMAL"
The primal solution is near integer optimal.

soltype
Solution types

"MSK_SOL_BAS"
The basic solution.

"MSK_SOL_ITR"
The interior solution.

"MSK_SOL_ITG"
The integer solution.

solveform
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.

stakey
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.

startpointtype
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 employed when choosing the bounds on the nonlinear variables. In particular very tight bounds should be avoided.

streamtype
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.

value
Integer values

"MSK_MAX_STR_LEN"
Maximum string length allowed in **MOSEK**.

"MSK_LICENSE_BUFFER_LENGTH"
The length of a license key buffer.

variabletype
Variable types

"MSK_VAR_TYPE_CONT"
Is a continuous variable.

"MSK_VAR_TYPE_INT"
Is an integer variable.

CHAPTER

# EIGHTEEN

# SUPPORTED FILE FORMATS

**MOSEK** supports a range of problem and solution formats listed in Table 18.1 and Table 18.2. The **Task format** is **MOSEK**'s native binary format and it supports all features that **MOSEK** supports. The **OPF format** is **MOSEK**'s human-readable alternative that supports nearly all features (everything except semidefinite problems). In general, text formats are significantly slower to read, but can be examined and edited directly in any text editor.

## Problem formats

See Table 18.1.

Table 18.1: List of supported file formats for optimization problems.

| Format Type | Ext. | Binary/Text | LP | QO | CQO | SDP |
|---|---|---|---|---|---|---|
| *LP* | lp | plain text | X | X | | |
| *MPS* | mps | plain text | X | X | | |
| *OPF* | opf | plain text | X | X | X | |
| *CBF* | cbf | plain text | X | | X | X |
| *OSiL* | xml | xml text | X | X | | |
| *Task format* | task | binary | X | X | X | X |
| *Jtask format* | jtask | text | X | X | X | X |

## Solution formats

See Table 18.2.

Table 18.2: List of supported solution formats.

| Format Type | Ext. | Binary/Text | Description |
|---|---|---|---|
| *SOL* | sol | plain text | Interior Solution |
| | bas | plain text | Basic Solution |
| | int | plain text | Integer |
| *Jsol format* | jsol | text | Solution |

## Compression

**MOSEK** supports GZIP compression of files. Problem files with an additional `.gz` extension are assumed to be compressed when read, and are automatically compressed when written. For example, a file called

```
problem.mps.gz
```

will be considered as a GZIP compressed MPS file.

## 18.1 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 on the form

$$
\begin{array}{lccccc}
\text{minimize/maximize} & & & c^T x + \frac{1}{2} q^o(x) & & \\
\text{subject to} & l^c & \leq & Ax + \frac{1}{2} q(x) & \leq & u^c, \\
 & l^x & \leq & x & \leq & u^x, \\
 & & & x_{\mathcal{J}} \text{ integer}, & &
\end{array}
$$

where

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

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

and it is assumed that

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

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

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

where it is assumed that

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

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

### 18.1.1 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.

$$c^T x + \frac{1}{2} x^T Q^o x.$$

The objective may be given a name by writing

```
myname:
```

before the expressions. If no name is given, then the objective is named `obj`.

The objective function contains linear and quadratic terms. The linear terms are written as:

```
4 x1 + x2 - 0.1 x3
```

and so forth. The quadratic terms are written in square brackets (`[ ]`) and are either squared or multiplied as in the examples

```
x1^2
```

and

```
x1 * x2
```

There may be zero or more pairs of brackets containing quadratic expressions.

An example of an objective section is

```
minimize
myobj: 4 x1 + x2 - 0.1 x3 + [ x1^2 + 2.1 x1 * x2 ]/2
```

Please note that the quadratic expressions are multiplied with $\frac{1}{2}$ , so that the above expression means

$$\text{minimize} \quad 4x_1 + x_2 - 0.1 \cdot x_3 + \tfrac{1}{2}(x_1^2 + 2.1 \cdot x_1 \cdot x_2)$$

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 `<=`) may be any of `<`, `<=`, `=`, `>`, `>=` (`<` and `<=` mean the same), and the bound may be any number.

In the standard LP format it is not possible to define more than one bound, but **MOSEK** supports defining ranged constraints by using double-colon (`::`) instead of a single-colon (`:`) after the constraint name, i.e.

$$-5 \leq x_1 + x_2 \leq 5 \tag{18.1}$$

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 (18.1) may be written as

$$x_1 + x_2 - sl_1 = 0, \ -5 \leq sl_1 \leq 5.$$

### 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
```

### 18.1.2 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
```

### 18.1.3 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.

#### MOSEK Extensions to the LP Format

Some optimization software packages employ a more strict definition of the LP format than the one used by **MOSEK**. The limitations imposed by the strict LP format are the following:

- Quadratic terms in the constraints are not allowed.
- Names can be only 16 characters long.
- Lines must not exceed 255 characters in length.

If an LP formatted file created by **MOSEK** should satisfy the strict definition, then the parameter

- *MSK_IPAR_WRITE_LP_STRICT_FORMAT*

should be set; note, however, that some problems cannot be written correctly as a strict LP formatted file. For instance, all names are truncated to 16 characters and hence they may loose their uniqueness and change the problem.

To get around some of the inconveniences converting from other problem formats, **MOSEK** allows lines to contain 1024 characters and names may have any length (shorter than the 1024 characters).

Internally in **MOSEK** names may contain any (printable) character, many of which cannot be used in LP names. Setting the parameters

- *MSK_IPAR_READ_LP_QUOTED_NAMES* and
- *MSK_IPAR_WRITE_LP_QUOTED_NAMES*

allows **MOSEK** to use quoted names. The first parameter tells **MOSEK** to remove quotes from quoted names e.g, "x1", when reading LP formatted files. The second parameter tells **MOSEK** to put quotes around any semi-illegal name (names beginning with a number or a period) and fully illegal name (containing illegal characters). As double quote is a legal character in the LP format, quoting semi-illegal names makes them legal in the pure LP format as long as they are still shorter than 16 characters. Fully illegal names are still illegal in a pure LP file.

### 18.1.4 The strict LP format

The LP format is not a formal standard and different vendors have slightly different interpretations of the LP format. To make **MOSEK**'s definition of the LP format more compatible with the definitions of other vendors, use the parameter setting

- *MSK_IPAR_WRITE_LP_STRICT_FORMAT* = *"MSK_ON"*

This setting may lead to truncation of some names and hence to an invalid LP file. The simple solution to this problem is to use the parameter setting

- *MSK_IPAR_WRITE_GENERIC_NAMES* = *"MSK_ON"*

which will cause all names to be renamed systematically in the output file.

### 18.1.5 Formatting of an LP File

A few parameters control the visual formatting of LP files written by **MOSEK** in order to make it easier to read the files. These parameters are

- *MSK_IPAR_WRITE_LP_LINE_WIDTH*
- *MSK_IPAR_WRITE_LP_TERMS_PER_LINE*

The first parameter sets the maximum number of characters on a single line. The default value is 80 corresponding roughly to the width of a standard text document.

The second parameter sets the maximum number of terms per line; a term means a sign, a coefficient, and a name (for example `+ 42 elephants`). The default value is 0, meaning that there is no maximum.

#### Unnamed Constraints

Reading and writing an LP file with **MOSEK** may change it superficially. If an LP file contains unnamed constraints or objective these are given their generic names when the file is read (however unnamed constraints in **MOSEK** are written without names).

## 18.2 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]*.

### 18.2.1 MPS File Structure

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

$$\begin{array}{ccccc} l^c & \leq & Ax + q(x) & \leq & u^c, \\ l^x & \leq & x & \leq & u^x, \\ & & x \in \mathcal{K}, & & \\ & & x_{\mathcal{J}} \ \ \text{integer}, & & \end{array} \tag{18.2}$$

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.

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

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

```
*        1         2         3         4         5         6
*234567890123456789012345678901234567890123456789012345678901234567890
NAME          [name]
OBJSENSE
[objsense]
OBJNAME
[objname]
ROWS
?  [cname1]
COLUMNS
[vname1]  [cname1]    [value1]     [vname3]  [value2]
RHS
[name]    [cname1]    [value1]     [cname2]  [value2]
RANGES
[name]    [cname1]    [value1]     [cname2]  [value2]
QSECTION      [cname1]
[vname1]  [vname2]    [value1]     [vname3]  [value2]
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

```
  [+|-]XXXXXXX.XXXXXX[[e|E][+|-]XXX]


where
```

```
.. code-block:: text

   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. 18.2.9 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.

### Section `NAME`

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. The OBJNAME section contains one line at most which has the form

```
objname
```

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 an unique name denoted by [cname1]. Please note that [cname1] starts in position 5 and the field can be at most 8 characters wide. An initial key ? must be present to specify the type of the constraint. The key can have the values E, G, L, or N with the following interpretation:

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

In the MPS format an objective vector is not specified explicitly, but one of the constraints having the key N will be used as the objective vector $c$ . In general, if multiple N type constraints are specified, then the first will be used as the objective vector $c$.

### 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 type | $l_i^c$ | $u_i^c$ |
|---|---|---|
| E | $v_1$ | $v_1$ |
| G | $v_1$ | |
| L | | $v_1$ |
| N | | |

An optional second element is specified by `[cname2]` and `[value2]` and is interpreted in the same way. Please note that it is not necessary to specify zero elements, because elements are assumed to be zero.

### 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 | | | |

### QSECTION (optional)

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

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

where the requirements for each field are:

| Field | Starting Position | 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 $Q^i_{kj}$ is assigned the value given by [value1] An optional second element is specified in the same way by the fields [vname1], [vname3], and [value2].

The example

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

has the following MPS file representation

```
* File: qo1.mps
NAME          qo1
ROWS
N  obj
G  c1
COLUMNS
```

```
x1        c1        1.0
x2        obj       -1.0
x2        c1        1.0
x3        c1        1.0
RHS
rhs       c1        1.0
QSECTION      obj
x1        x1        2.0
x1        x3        -1.0
x2        x2        0.2
x3        x3        2.0
ENDATA
```

Regarding the 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 It stores all the nonzeros coefficients, withouot taking advantage of the symmetry of the $Q$ matrix.
- QUADOBJ It only store 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 apper 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 $1/2x^TQx$.

The example

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

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.

### 18.2.2 `QCMATRIX` (optional)

A `QCMATRIX` section allows to specify the quadratic part of a given constraints. Within the `QCMATRIX` 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 $Q^i_{kj}$ is assigned the value given by `[value1]`. Moreover, the quadratic term is represented as $1/2x^TQx$.

The example

$$\begin{array}{lccl} \text{minimize} & x_2 & & \\ \text{subject to} & x_1 + x_2 + x_3 & \geq & 1, \\ & \frac{1}{2}(-2x_1x_3 + 0.2x_2^2 + 2x_3^2) & \leq & 10, \\ & x \geq 0 & & \end{array}$$

has the following MPS file representation

```
* File: qo1.mps
NAME          qo1
ROWS
 N  obj
 G  c1
 L  q1
COLUMNS
    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
    x1        x1        2.0
    x1        x3        -1.0
    x3        x1        -1.0
    x2        x2        0.2
    x3        x3        2.0
ENDATA
```

Regarding the `QCMATRIX`s please note that:

- Only one `QCMATRIX` is allowed for each constraint.
- The `QCMATRIX`s can appear in an arbitrary order after the `COLUMNS` section.
- All variable names occurring in the `QSECTION` must already be specified in the `COLUMNS` section.
- A `QCMATRIX` does not exploit the symmetry of $Q$: an off-diagonal entry $(i, j)$ should apperas twice.

### 18.2.3 `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 which bounds are modified by the record. ?? and [value1] are used to modify the bound vectors according to the following table:

| ?? | $l_j^x$ | $u_j^x$ | Made integer (added to $\mathcal{J}$ ) |
|---|---|---|---|
| FR | $-\infty$ | $\infty$ | No |
| FX | $v_1$ | $v_1$ | No |
| LO | $v_1$ | unchanged | No |
| MI | $-\infty$ | unchanged | No |
| PL | unchanged | $\infty$ | No |
| UP | unchanged | $v_1$ | No |
| BV | 0 | 1 | Yes |
| LI | $\lceil v_1 \rceil$ | unchanged | Yes |
| UI | unchanged | $\lfloor v_1 \rfloor$ | Yes |

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

### 18.2.4 CSECTION (optional)

The purpose of the CSECTION is to specify the constraint

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

in (18.2). It is assumed that $\mathcal{K}$ satisfies the following requirements. Let

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

be vectors comprised of parts of the decision variables $x$ so that each decision variable is a member of exactly **one** vector $x^t$, for example

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

Next define

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

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

- $\mathbb{R}$ set:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} \right\}.$$

- Quadratic cone:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\}. \tag{18.3}$$

- Rotated quadratic cone:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : 2x_1 x_2 \geq \sum_{j=3}^{n^t} x_j^2, \quad x_1, x_2 \geq 0 \right\}. \tag{18.4}$$

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

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

$$x_4 \geq \sqrt{x_5^2 + x_8^2}$$

and the rotated quadratic cone

$$x_3 x_7 \geq x_1^2 + x_0^2, \quad x_3, x_7 \geq 0,$$

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
*23456789012345678901234567890123456789012345678901234567890
CSECTION      konea       0.0          QUAD
x4
x5
x8
CSECTION      koneb       0.0          RQUAD
x7
x3
x1
x0
```

This first CSECTION specifies the cone (18.3) which is given the name konea. This is a quadratic cone which is specified by the keyword QUAD in the CSECTION header. The 0.0 value in the CSECTION header is not used by the QUAD cone.

The second CSECTION specifies the rotated quadratic cone (18.4). Please note the keyword RQUAD in the CSECTION which is used to specify that the cone is a rotated quadratic cone instead of a quadratic cone. The 0.0 value in the CSECTION header is not used by the RQUAD cone.

In general, a CSECTION header has the format

```
CSECTION      [kname1]     [value1]     [ktype]
```

where the requirement for each field are as follows:

| Field | Starting Position | Max Width | Required | Description |
|---|---|---|---|---|
| [kname1] | 5 | 8 | Yes | Name of the cone |
| [value1] | 15 | 12 | No | Cone parameter |
| [ktype] | 25 | | Yes | Type of the cone. |

The possible cone type keys are:

| Cone type key | Members | Interpretation. |
|---|---|---|
| QUAD | $\leq 1$ | Quadratic cone i.e. (18.3). |
| RQUAD | $\leq 2$ | Rotated quadratic cone i.e. (18.4). |

Please note that a quadratic cone must have at least one member whereas a rotated quadratic cone must have at least two members. A record in the CSECTION has the format

```
[vname1]
```

where the requirements for each field are

| Field | Starting Position | Max Width | required | Description |
|---|---|---|---|---|
| [vname1] | 2 | 8 | Yes | A valid variable name |

The most important restriction with respect to the `CSECTION` is that a variable must occur in only one `CSECTION`.

### 18.2.5 ENDATA

This keyword denotes the end of the MPS file.

### 18.2.6 Integer Variables

Using special bound keys in the `BOUNDS` section it is possible to specify that some or all of the variables should be integer-constrained i.e. be members of $\mathcal{J}$ . However, an alternative method is available.

This method is available only for backward compatibility and we recommend that it is not used. This method requires that markers are placed in the `COLUMNS` section as in the example:

```
COLUMNS
x1        obj       -10.0          c1        0.7
x1        c2        0.5            c3        1.0
x1        c4        0.1
* Start of integer-constrained variables.
MARK000   'MARKER'                 'INTORG'
x2        obj       -9.0           c1        1.0
x2        c2        0.8333333333   c3        0.66666667
x2        c4        0.25
x3        obj       1.0            c6        2.0
MARK001   'MARKER'                 'INTEND'
```

- End of integer-constrained variables.

Please note that special marker lines are used to indicate the start and the end of the integer variables. Furthermore be aware of the following

- IMPORTANT: All variables between the markers are assigned a default lower bound of 0 and a default upper bound of 1. **This may not be what is intended.** If it is not intended, the correct bounds should be defined in the `BOUNDS` section of the MPS formatted file.
- **MOSEK** ignores field 1, i.e. `MARK0001` and `MARK001`, however, other optimization systems require them.
- Field 2, i.e. `MARKER`, must be specified including the single quotes. This implies that no row can be assigned the name `MARKER`.
- Field 3 is ignored and should be left blank.
- Field 4, i.e. `INTORG` and `INTEND`, must be specified.
- It is possible to specify several such integer marker sections within the `COLUMNS` section.

### 18.2.7 General Limitations

- An MPS file should be an ASCII file.

### 18.2.8 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.

### 18.2.9 The Free MPS Format

**MOSEK** supports a free format variation of the MPS format. The free format is similar to the MPS file format but less restrictive, e.g. it allows longer names. However, it also presents two main limitations:

- A name must not contain any blanks.
- By default a line in the MPS file must not contain more than 1024 characters. However, by modifying the parameter *MSK_IPAR_READ_MPS_WIDTH* an arbitrary large line width will be accepted.

To use the free MPS format instead of the default MPS format the **MOSEK** parameter *MSK_IPAR_READ_MPS_FORMAT* should be changed.

## 18.3 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.

### 18.3.1 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 in quotes [/tag]
```

Unnamed arguments are identified by their order, while named arguments may appear in any order, but never before an unnamed argument. The `value` can be a quoted, single-quoted or double-quoted text string, i.e.

```
[tag 'value']     single-quoted value [/tag]
[tag arg='value'] single-quoted value [/tag]
[tag "value"]     double-quoted value [/tag]
[tag arg="value"] double-quoted value [/tag]
```

### Sections

The recognized tags are

#### [comment]

A comment section. This can contain *almost* any text: Between single quotes (') or double quotes (") any text may appear. Outside quotes the markup characters ([ and ]) must be prefixed by backslashes. Both single and double quotes may appear alone or inside a pair of quotes if it is prefixed by a backslash.

#### [objective]

The objective function: This accepts one or two parameters, where the first one (in the above example `min`) is either `min` or `max` (regardless of case) and defines the objective sense, and the second one (above `myobj`), if present, is the objective name. The section may contain linear and quadratic expressions. If several objectives are specified, all but the last are ignored.

#### [constraints]

This does not directly contain any data, but may contain the subsection `con` defining a linear constraint.

`[con]` defines a single constraint; if an argument is present (`[con NAME]`) this is used as the name of the constraint, otherwise it is given a null-name. The section contains a constraint definition written as linear and quadratic expressions with a lower bound, an upper bound, with both or with an equality. Examples:

```
[constraints]
[con 'con1'] 0 <= x + y       [/con]
[con 'con2'] 0 >= x + y       [/con]
[con 'con3'] 0 <= x + y <= 10 [/con]
[con 'con4']      x + y  = 10 [/con]
[/constraints]
```

Constraint names are unique. If a constraint is specified which has the same name as a previously defined constraint, the new constraint replaces the existing one.

### [bounds]

This does not directly contain any data, but may contain the subsections `b` (linear bounds on variables) and `cone` (quadratic cone).

`[b]`. Bound definition on one or several variables separated by comma (`,`). An upper or lower bound on a variable replaces any earlier defined bound on that variable. If only one bound (upper or lower) is given only this bound is replaced. This means that upper and lower bounds can be specified separately. So the OPF bound definition:

```
[b]  x,y >= -10  [/b]
[b]  x,y <= 10   [/b]
```

results in the bound $-10 \leq x, y \leq 10$.

`[cone]`. currently supports the *quadratic cone* and the *rotated quadratic cone*.

A conic constraint is defined as a set of variables which belong to a single unique cone.

- A quadratic cone of $n$ variables $x_1, \ldots, x_n$ defines a constraint of the form

$$x_1^2 \geq \sum_{i=2}^{n} x_i^2, \quad x_1 \geq 0.$$

- A rotated quadratic cone of $n$ variables $x_1, \ldots, x_n$ defines a constraint of the form

$$2x_1x_2 \geq \sum_{i=3}^{n} x_i^2, \quad x_1, x_2 \geq 0.$$

A `[bounds]`-section example:

```
[bounds]
[b]  0 <= x,y <= 10  [/b] # ranged bound
[b] 10 >= x,y >=  0  [/b] # ranged bound
[b]  0 <= x,y <= inf [/b] # using inf
[b]       x,y free   [/b] # free variables
# Let (x,y,z,w) belong to the cone K
[cone quad]  x,y,z,w  [/cone] # quadratic cone
[cone rquad] x,y,z,w  [/cone] # rotated quadratic cone
[/bounds]
```

By default all variables are free.

### [variables]

This defines an ordering of variables as they should appear in the problem. This is simply a space-separated list of variable names. Optionally, an attribute can be added `[variables disallow_new_variables]` indicating that if any variable not listed here occurs later in the file it is an error.

### [integer]

This contains a space-separated list of variables and defines the constraint that the listed variables must be integer values.

### [hints]

This may contain only non-essential data; for example estimates of the number of variables, constraints and non-zeros. Placed before all other sections containing data this may reduce the time spent reading the file.

In the `hints` section, any subsection which is not recognized by **MOSEK** is simply ignored. In this section a hint in a subsection is defined as follows:

```
[hint ITEM] value [/hint]
```

where `ITEM` may be replaced by `numvar` (number of variables), `numcon` (number of linear/quadratic constraints), `numanz` (number of linear non-zeros in constraints) and `numqnz` (number of quadratic non-zeros in constraints).

### [solutions]

This section can contain a set of full or partial solutions to a problem. Each solution must be specified using a `[solution]`-section, i.e.

```
[solutions]
[solution]...[/solution] #solution 1
[solution]...[/solution] #solution 2
#other solutions....
[solution]...[/solution] #solution n
[/solutions]
```

Note that a `[solution]`-section must be always specified inside a `[solutions]`-section. The syntax of a `[solution]`-section is the following:

```
[solution SOLTYPE status=STATUS]...[/solution]
```

where `SOLTYPE` is one of the strings

- `interior`, a non-basic solution,
- `basic`, a basic solution,
- `integer`, an integer solution,

and `STATUS` is one of the strings

- `UNKNOWN`,
- `OPTIMAL`,
- `INTEGER_OPTIMAL`,
- `PRIM_FEAS`,
- `DUAL_FEAS`,
- `PRIM_AND_DUAL_FEAS`,
- `NEAR_OPTIMAL`,
- `NEAR_PRIM_FEAS`,
- `NEAR_DUAL_FEAS`,
- `NEAR_PRIM_AND_DUAL_FEAS`,
- `PRIM_INFEAS_CER`,
- `DUAL_INFEAS_CER`,
- `NEAR_PRIM_INFEAS_CER`,

- NEAR_DUAL_INFEAS_CER,
- NEAR_INTEGER_OPTIMAL.

Most of these values are irrelevant for input solutions; when constructing a solution for simplex hot-start or an initial solution for a mixed integer problem the safe setting is UNKNOWN.

A [solution]-section contains [con] and [var] sections. Each [con] and [var] section defines solution information for a single variable or constraint, specified as list of KEYWORD/value pairs, in any order, written as

```
KEYWORD=value
```

Allowed keywords are as follows:

- sk. The status of the item, where the value is one of the following strings:
  - LOW, the item is on its lower bound.
  - UPR, the item is on its upper bound.
  - FIX, it is a fixed item.
  - BAS, the item is in the basis.
  - SUPBAS, the item is super basic.
  - UNK, the status is unknown.
  - INF, the item is outside its bounds (infeasible).
- lvl Defines the level of the item.
- sl Defines the level of the dual variable associated with its lower bound.
- su Defines the level of the dual variable associated with its upper bound.
- sn Defines the level of the variable associated with its cone.
- y Defines the level of the corresponding dual variable (for constraints only).

A [var] section should always contain the items sk, lvl, sl and su. Items sl and su are not required for integer solutions.

A [con] section should always contain sk, lvl, sl, su and y.

An example of a solution section

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

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

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

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

### 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]+)?
```

### 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"
```

## 18.3.2 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]
```

### 18.3.3 Writing OPF Files from MOSEK

To write an OPF file set the parameter *MSK_IPAR_WRITE_DATA_FORMAT* to *"MSK_DATA_FORMAT_OP"* as this ensures that OPF format is used.

Then modify the following parameters to define what the file should contain:

| | |
|---|---|
| *MSK_IPAR_OPF_WRITE_SOL_BAS* | Include basic solution, if defined. |
| *MSK_IPAR_OPF_WRITE_SOL_ITG* | Include integer solution, if defined. |
| *MSK_IPAR_OPF_WRITE_SOL_ITR* | Include interior solution, if defined. |
| *MSK_IPAR_OPF_WRITE_SOLUTIONS* | Include solutions if they are defined. If this is off, no solutions are included. |
| *MSK_IPAR_OPF_WRITE_HEADER* | Include a small header with comments. |
| *MSK_IPAR_OPF_WRITE_PROBLEM* | Include the problem itself — objective, constraints and bounds. |
| *MSK_IPAR_OPF_WRITE_PARAMETERS* | Include all parameter settings. |
| *MSK_IPAR_OPF_WRITE_HINTS* | Include hints about the size of the problem. |

### 18.3.4 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:

$$\begin{array}{lccccccccc}
\text{maximize} & 3x_0 & + & 1x_1 & + & 5x_2 & + & 1x_3 & & \\
\text{subject to} & 3x_0 & + & 1x_1 & + & 2x_2 & & & = & 30, \\
 & 2x_0 & + & 1x_1 & + & 3x_2 & + & 1x_3 & \geq & 15, \\
 & & & 2x_1 & & & + & 3x_3 & \leq & 25,
\end{array}$$

having the bounds

$$\begin{array}{ccccc}
0 & \leq & x_0 & \leq & \infty, \\
0 & \leq & x_1 & \leq & 10, \\
0 & \leq & x_2 & \leq & \infty, \\
0 & \leq & x_3 & \leq & \infty.
\end{array}$$

In the OPF format the example is displayed as shown in Listing 18.1.

Listing 18.1: 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

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

This can be formulated in `opf` as shown below.

Listing 18.2: 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]

[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:

$$
\begin{array}{lccc}
\text{minimize} & x_3 + x_4 + x_5 & & \\
\text{subject to} & x_0 + x_1 + 2x_2 & = & 1, \\
 & x_0, x_1, x_2 & \geq & 0, \\
 & x_3 \geq \sqrt{x_0^2 + x_1^2}, & & \\
 & 2x_4 x_5 \geq x_2^2. & &
\end{array}
$$

Please note that the type of the cones is defined by the parameter to `[cone ...]`; the content of the `cone`-section is the names of variables that belong to the cone. The resulting OPF file is in Listing 18.3.

Listing 18.3: 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
  [b] x4,x5,x6 free [/b]

  # Define quadratic cone: x4 >= sqrt( x1^2 + x2^2 )
  [cone quad 'k1'] x4, x1, x2 [/cone]

  # Define rotated quadratic cone: 2 x5 x6 >= x3^2
  [cone rquad 'k2'] x5, x6, x3 [/cone]
[/bounds]
```

### Mixed Integer Example `milo1.opf`

Consider the mixed integer problem:

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

This can be implemented in `OPF` with the file in Listing 18.4.

Listing 18.4: 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]
```

## 18.4 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) 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 problem structure is separated from the problem data, and the format moreover facilitates benchmarking of hotstart capability through sequences of changes.

### 18.4.1 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:

$$\begin{array}{lll} \min / \max & g^{obj} & \\ & g_i \in \mathcal{K}_i, & i \in \mathcal{I}, \\ \text{s.t.} & G_i \in \mathcal{K}_i, & i \in \mathcal{I}^{PSD}, \\ & x_j \in \mathcal{K}_j, & j \in \mathcal{J}, \\ & \overline{X}_j \in \mathcal{K}_j, & j \in \mathcal{J}^{PSD}. \end{array} \tag{18.5}$$

- **Variables** are either scalar variables, $x_j$ for $j \in \mathcal{J}$, or 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 $i \in \mathcal{I}^{PSD}$

$$g_i = \sum_{j \in \mathcal{J}^{PSD}} \langle F_{ij}, X_j \rangle + \sum_{j \in \mathcal{J}} a_{ij} x_j + b_i,$$
$$G_i = \sum_{j \in \mathcal{J}} x_j H_{ij} + D_i.$$

- The **objective function** is a scalar-valued affine expression of the variables, either to be minimized or maximized. We refer to this expression as $g^{obj}$

$$g^{obj} = \sum_{j \in \mathcal{J}^{PSD}} \langle F_j^{obj}, X_j \rangle + \sum_{j \in \mathcal{J}} a_j^{obj} x_j + b^{obj}.$$

CBF format can represent the following 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

$$\{x \in \mathbb{R}^n \mid x_j \geq 0 \text{ for } j = 1, \ldots, n\}, \text{ for } n \geq 1.$$

- **Negative orthant** - A cone in the linear family defined by

$$\{x \in \mathbb{R}^n \mid x_j \leq 0 \text{ for } j = 1, \ldots, n\}, \text{ for } n \geq 1.$$

- **Fixpoint zero** - A cone in the linear family defined by

$$\{x \in \mathbb{R}^n \mid x_j = 0 \text{ for } j = 1, \ldots, n\}, \text{ for } n \geq 1.$$

- **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

$$\left\{ \begin{pmatrix} p \\ q \\ x \end{pmatrix} \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^{n-2},\ 2pq \geq x^T x,\ p \geq 0,\ q \geq 0 \right\}, \text{ for } n \geq 3.$$

## 18.4.2 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.

#### Embedded hotstart-sequences

A sequence of problem instances, based on the same problem structure, is within a single file. This is facilitated via the CHANGE within the problem data information group, as a separator between the information items of each instance. The information items following a CHANGE keyword are appending to, or changing (e.g., setting coefficients back to their default value of zero), the problem data of the preceding instance.

The sequence is intended for benchmarking of hotstart capability, where the solvers can reuse their internal state and solution (subject to the achieved accuracy) as warmpoint for the succeeding instance. Whenever this feature is unsupported or undesired, the keyword CHANGE should be interpreted as the end of file.

#### 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 seperator 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.

### 18.4.3 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, $\{0, 1, \ldots\}$, 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

  $$x \in \mathcal{K}_1^{n_1} \times \mathcal{K}_2^{n_2} \times \cdots \times \mathcal{K}_k^{n_k}$$

  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 18.3. 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

  $$g \in \mathcal{K}_1^{m_1} \times \mathcal{K}_2^{m_2} \times \cdots \times \mathcal{K}_k^{m_k}$$

  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 18.3.

- **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 minimum sizes are given as follows.

Table 18.3: Cones available in the CBF format

| Name | CBF keyword | Cone family |
|---|---|---|
| Free domain | `F` | linear |
| Positive orthant | `L+` | linear |
| Negative orthant | `L-` | linear |
| Fixpoint zero | `L=` | linear |
| Quadratic cone | `Q` | second-order |
| Rotated quadratic cone | `QR` | second-order |

### 18.4.4 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.

#### OBJSENSE

*Description*: Define the objective sense.

`HEADER`: None

`BODY`: One line formatted as:

```
STR
```

having `MIN` indicates minimize, and `MAX` indicates maximize. Capital 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 18.3), 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 18.3), 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.

### ACOORD

*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.

#### CHANGE

Start of a new instance specification based on changes to the previous. Can be interpreted as the end of file when the hotstart-sequence is unsupported or undesired.

`BODY`: None

`Header`: None

### 18.4.5 CBF Format Examples

#### Minimal Working Example

The conic optimization problem (18.6) , has three variables in a quadratic cone - first one is integer - and an affine expression in domain 0 (equality constraint).

$$
\begin{array}{ll}
\text{minimize} & 5.1\,x_0 \\
\text{subject to} & 6.2\,x_1 + 7.3\,x_2 - 8.4 \in \{0\} \\
& x \in \mathcal{Q}^3,\ x_0 \in \mathbb{Z}.
\end{array}
\tag{18.6}
$$

Its formulation in the Conic Benchmark Format begins with the version of the CBF format used, to safeguard against later revisions.

```
VER
1
```

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.

```
OBJACOORD
1
0 5.1

ACOORD
2
0 1 6.2
0 2 7.3

BCOORD
1
0 -8.4
```

This concludes the example.

### Mixing Linear, Second-order and Semidefinite Cones

The conic optimization problem (18.7), has a semidefinite cone, a quadratic cone over unordered subindices, and two equality constraints.

$$
\begin{array}{lllll}
\text{minimize} & \left\langle \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}, X_1 \right\rangle + x_1 & & \\
\text{subject to} & \left\langle \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, X_1 \right\rangle + x_1 & = & 1.0\,, \\
& \left\langle \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, X_1 \right\rangle + x_0 + x_2 & = & 0.5\,, \\
& x_1 \geq \sqrt{x_0^2 + x_2^2}\,, & & \\
& X_1 \succeq \mathbf{0}\,. & &
\end{array}
\tag{18.7}
$$

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 1.
VER
1

# 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
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
OBJACOORD
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...
ACOORD
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 in.

$$
\begin{array}{lllll}
\text{minimize} & \left\langle \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, X_1 \right\rangle + x_1 + x_2 + 1 & & \\
\text{subject to} & \left\langle \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, X_1 \right\rangle - x_1 - x_2 & \geq & 0.0\,, \\
& x_1 \begin{bmatrix} 0 & 1 \\ 1 & 3 \end{bmatrix} + x_2 \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \succeq & \mathbf{0}\,, \\
& X_1 \succeq \mathbf{0}\,. & &
\end{array}
\tag{18.8}
$$

Its formulation in the CBF format is written in what follows

```
# File written using this version of the Conic Benchmark Format:
#     | Version 1.
VER
1

# 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:
#     | a^{obj}[0] = 1.0
#     | a^{obj}[1] = 1.0
OBJACOORD
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
ACOORD
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
```

```
DCOORD
2
0 0 0 -1.0
0 1 1 -1.0
```

### Optimization Over a Sequence of Objectives

The linear optimization problem (18.9), is defined for a sequence of objectives such that hotstarting from one to the next might be advantages.

$$\begin{array}{lccl} \text{maximize}_k & g_k^{obj} & & \\ \text{subject to} & 50\,x_0 + 31 & \leq & 250\,, \\ & 3\,x_0 - 2x_1 & \geq & -4\,, \\ & x \in \mathbb{R}_+^2, & & \end{array} \tag{18.9}$$

given,

1. $g_0^{obj} = x_0 + 0.64x_1$.
2. $g_1^{obj} = 1.11x_0 + 0.76x_1$.
3. $g_2^{obj} = 1.11x_0 + 0.85x_1$.

Its formulation in the CBF format is reported in Listing 18.5.

Listing 18.5: Problem (18.9) in CBF format.

```
# File written using this version of the Conic Benchmark Format:
#     | Version 1.
VER
1

# The sense of the objective is:
#     | Maximize.
OBJSENSE
MAX

# Two scalar variables in this one conic domain:
#     | Two are nonnegative.
VAR
2 1
L+ 2

# Two scalar constraints with affine expressions in these two conic domains:
#     | One is in the nonpositive domain.
#     | One is in the nonnegative domain.
CON
2 2
L- 1
L+ 1

# Two coordinates in a^{obj}_j coefficients:
#     | a^{obj}[0] = 1.0
#     | a^{obj}[1] = 0.64
OBJACOORD
2
0 1.0
1 0.64

# Four coordinates in a_ij coefficients:
#     | a[0,0] = 50.0
#     | a[1,0] = 3.0
```

```
#     | and more...
ACOORD
4
0 0 50.0
1 0 3.0
0 1 31.0
1 1 -2.0

# Two coordinates in b_i coefficients:
#     | b[0] = -250.0
#     | b[1] = 4.0
BCOORD
2
0 -250.0
1 4.0

# New problem instance defined in terms of changes.
CHANGE

# Two coordinate changes in a^{obj}_j coefficients. Now it is:
#     | a^{obj}[0] = 1.11
#     | a^{obj}[1] = 0.76
OBJACOORD
2
0 1.11
1 0.76

# New problem instance defined in terms of changes.
CHANGE

# One coordinate change in a^{obj}_j coefficients. Now it is:
#     | a^{obj}[0] = 1.11
#     | a^{obj}[1] = 0.85
OBJACOORD
1
1 0.85
```

## 18.5 The XML (OSiL) Format

**MOSEK** can write data in the standard OSiL xml format. For a definition of the OSiL format please see http://www.optimizationservices.org/.

Only linear constraints (possibly with integer variables) are supported. By default output files with the extension `.xml` are written in the OSiL format.

The parameter *MSK_IPAR_WRITE_XML_MODE* controls if the linear coefficients in the $A$ matrix are written in row or column order.

## 18.6 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 quadratic, 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:

- The task format *does not* support General Convex problems since these are defined by arbitrary user-defined functions.
- 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.

## 18.7 The JSON Format

**MOSEK** provides the possibility to read/write problems in valid JSON format.

> JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

The official JSON website http://www.json.org provides plenty of information along with the format definition.

**MOSEK** defines two JSON-like formats:

- *jtask*
- *jsol*

**Warning:** Despite being text-based human-readable formats, *jtask* and *jsol* files will include no indentation and no new-lines, in order to keep the files as compact as possible. We therefore strongly advise to use JSON viewer tools to inspect *jtask* and *jsol* files.

### 18.7.1 *jtask* format

It stores a problem instance. The *jtask* format contains the same information as a *task format*.

Even though a *jtask* file is human-readable, we do not recommend users to create it by hand, but to rely on **MOSEK**.

### 18.7.2 *jsol* format

It stores a problem solution. The *jsol* format contains all solutions and information items.

### 18.7.3 A *jtask* example

In Listing 18.6 we present a file in the *jtask* format that corresponds to the sample problem from `lo1.lp`. The listing has been formatted for readability.

Listing 18.6: A formatted *jtask* file for the `lo1.lp` example.


```
{
    "$schema":"http://mosek.com/json/schema#",
    "Task/INFO":{
        "taskname":"lo1",
        "numvar":4,
        "numcon":3,
        "numcone":0,
        "numbarvar":0,
        "numanz":9,
        "numsymmat":0,
        "mosekver":[
            8,
            0,
            0,
            9
        ]
    },
    "Task/data":{
        "var":{
            "name":[
                "x1",
                "x2",
                "x3",
                "x4"
            ],
            "bk":[
                "lo",
                "ra",
                "lo",
                "lo"
            ],
            "bl":[
                0.0,
                0.0,
                0.0,
                0.0
            ],
            "bu":[
                1e+30,
                1e+1,
                1e+30,
                1e+30
            ],
            "type":[
                "cont",
                "cont",
                "cont",
                "cont"
            ]
        },
        "con":{
            "name":[
                "c1",
                "c2",
                "c3"
            ],
            "bk":[
                "fx",
                "lo",
                "up"
```

```
            ],
            "bl":[
                3e+1,
                1.5e+1,
                    -1e+30
            ],
            "bu":[
                3e+1,
                1e+30,
                2.5e+1
            ]
        },
        "objective":{
            "sense":"max",
            "name":"obj",
            "c":{
                "subj":[
                    0,
                    1,
                    2,
                    3
                ],
                "val":[
                    3e+0,
                    1e+0,
                    5e+0,
                    1e+0
                ]
            },
            "cfix":0.0
        },
        "A":{
            "subi":[
                0,
                0,
                0,
                1,
                1,
                1,
                1,
                2,
                2
            ],
            "subj":[
                0,
                1,
                2,
                0,
                1,
                2,
                3,
                1,
                3
            ],
            "val":[
                3e+0,
                1e+0,
                2e+0,
                2e+0,
                1e+0,
                3e+0,
                1e+0,
                2e+0,
```

```
                3e+0
            ]
        }
    },
    "Task/parameters":{
        "iparam":{
            "ANA_SOL_BASIS":"ON",
            "ANA_SOL_PRINT_VIOLATED":"OFF",
            "AUTO_SORT_A_BEFORE_OPT":"OFF",
            "AUTO_UPDATE_SOL_INFO":"OFF",
            "BASIS_SOLVE_USE_PLUS_ONE":"OFF",
            "BI_CLEAN_OPTIMIZER":"OPTIMIZER_FREE",
            "BI_IGNORE_MAX_ITER":"OFF",
            "BI_IGNORE_NUM_ERROR":"OFF",
            "BI_MAX_ITERATIONS":1000000,
            "CACHE_LICENSE":"ON",
            "CHECK_CONVEXITY":"CHECK_CONVEXITY_FULL",
            "COMPRESS_STATFILE":"ON",
            "CONCURRENT_NUM_OPTIMIZERS":2,
            "CONCURRENT_PRIORITY_DUAL_SIMPLEX":2,
            "CONCURRENT_PRIORITY_FREE_SIMPLEX":3,
            "CONCURRENT_PRIORITY_INTPNT":4,
            "CONCURRENT_PRIORITY_PRIMAL_SIMPLEX":1,
            "FEASREPAIR_OPTIMIZE":"FEASREPAIR_OPTIMIZE_NONE",
            "INFEAS_GENERIC_NAMES":"OFF",
            "INFEAS_PREFER_PRIMAL":"ON",
            "INFEAS_REPORT_AUTO":"OFF",
            "INFEAS_REPORT_LEVEL":1,
            "INTPNT_BASIS":"BI_ALWAYS",
            "INTPNT_DIFF_STEP":"ON",
            "INTPNT_FACTOR_DEBUG_LVL":0,
            "INTPNT_FACTOR_METHOD":0,
            "INTPNT_HOTSTART":"INTPNT_HOTSTART_NONE",
            "INTPNT_MAX_ITERATIONS":400,
            "INTPNT_MAX_NUM_COR":-1,
            "INTPNT_MAX_NUM_REFINEMENT_STEPS":-1,
            "INTPNT_OFF_COL_TRH":40,
            "INTPNT_ORDER_METHOD":"ORDER_METHOD_FREE",
            "INTPNT_REGULARIZATION_USE":"ON",
            "INTPNT_SCALING":"SCALING_FREE",
            "INTPNT_SOLVE_FORM":"SOLVE_FREE",
            "INTPNT_STARTING_POINT":"STARTING_POINT_FREE",
            "LIC_TRH_EXPIRY_WRN":7,
            "LICENSE_DEBUG":"OFF",
            "LICENSE_PAUSE_TIME":0,
            "LICENSE_SUPPRESS_EXPIRE_WRNS":"OFF",
            "LICENSE_WAIT":"OFF",
            "LOG":10,
            "LOG_ANA_PRO":1,
            "LOG_BI":4,
            "LOG_BI_FREQ":2500,
            "LOG_CHECK_CONVEXITY":0,
            "LOG_CONCURRENT":1,
            "LOG_CUT_SECOND_OPT":1,
            "LOG_EXPAND":0,
            "LOG_FACTOR":1,
            "LOG_FEAS_REPAIR":1,
            "LOG_FILE":1,
            "LOG_HEAD":1,
            "LOG_INFEAS_ANA":1,
            "LOG_INTPNT":4,
            "LOG_MIO":4,
            "LOG_MIO_FREQ":1000,
```

```
            "LOG_OPTIMIZER":1,
            "LOG_ORDER":1,
            "LOG_PRESOLVE":1,
            "LOG_RESPONSE":0,
            "LOG_SENSITIVITY":1,
            "LOG_SENSITIVITY_OPT":0,
            "LOG_SIM":4,
            "LOG_SIM_FREQ":1000,
            "LOG_SIM_MINOR":1,
            "LOG_STORAGE":1,
            "MAX_NUM_WARNINGS":10,
            "MIO_BRANCH_DIR":"BRANCH_DIR_FREE",
            "MIO_CONSTRUCT_SOL":"OFF",
            "MIO_CUT_CLIQUE":"ON",
            "MIO_CUT_CMIR":"ON",
            "MIO_CUT_GMI":"ON",
            "MIO_CUT_KNAPSACK_COVER":"OFF",
            "MIO_HEURISTIC_LEVEL":-1,
            "MIO_MAX_NUM_BRANCHES":-1,
            "MIO_MAX_NUM_RELAXS":-1,
            "MIO_MAX_NUM_SOLUTIONS":-1,
            "MIO_MODE":"MIO_MODE_SATISFIED",
            "MIO_MT_USER_CB":"ON",
            "MIO_NODE_OPTIMIZER":"OPTIMIZER_FREE",
            "MIO_NODE_SELECTION":"MIO_NODE_SELECTION_FREE",
            "MIO_PERSPECTIVE_REFORMULATE":"ON",
            "MIO_PROBING_LEVEL":-1,
            "MIO_RINS_MAX_NODES":-1,
            "MIO_ROOT_OPTIMIZER":"OPTIMIZER_FREE",
            "MIO_ROOT_REPEAT_PRESOLVE_LEVEL":-1,
            "MT_SPINCOUNT":0,
            "NUM_THREADS":0,
            "OPF_MAX_TERMS_PER_LINE":5,
            "OPF_WRITE_HEADER":"ON",
            "OPF_WRITE_HINTS":"ON",
            "OPF_WRITE_PARAMETERS":"OFF",
            "OPF_WRITE_PROBLEM":"ON",
            "OPF_WRITE_SOL_BAS":"ON",
            "OPF_WRITE_SOL_ITG":"ON",
            "OPF_WRITE_SOL_ITR":"ON",
            "OPF_WRITE_SOLUTIONS":"OFF",
            "OPTIMIZER":"OPTIMIZER_FREE",
            "PARAM_READ_CASE_NAME":"ON",
            "PARAM_READ_IGN_ERROR":"OFF",
            "PRESOLVE_ELIMINATOR_MAX_FILL":-1,
            "PRESOLVE_ELIMINATOR_MAX_NUM_TRIES":-1,
            "PRESOLVE_LEVEL":-1,
            "PRESOLVE_LINDEP_ABS_WORK_TRH":100,
            "PRESOLVE_LINDEP_REL_WORK_TRH":100,
            "PRESOLVE_LINDEP_USE":"ON",
            "PRESOLVE_MAX_NUM_REDUCTIONS":-1,
            "PRESOLVE_USE":"PRESOLVE_MODE_FREE",
            "PRIMAL_REPAIR_OPTIMIZER":"OPTIMIZER_FREE",
            "QO_SEPARABLE_REFORMULATION":"OFF",
            "READ_DATA_COMPRESSED":"COMPRESS_FREE",
            "READ_DATA_FORMAT":"DATA_FORMAT_EXTENSION",
            "READ_DEBUG":"OFF",
            "READ_KEEP_FREE_CON":"OFF",
            "READ_LP_DROP_NEW_VARS_IN_BOU":"OFF",
            "READ_LP_QUOTED_NAMES":"ON",
            "READ_MPS_FORMAT":"MPS_FORMAT_FREE",
            "READ_MPS_WIDTH":1024,
            "READ_TASK_IGNORE_PARAM":"OFF",
```

```
            "SENSITIVITY_ALL":"OFF",
            "SENSITIVITY_OPTIMIZER":"OPTIMIZER_FREE_SIMPLEX",
            "SENSITIVITY_TYPE":"SENSITIVITY_TYPE_BASIS",
            "SIM_BASIS_FACTOR_USE":"ON",
            "SIM_DEGEN":"SIM_DEGEN_FREE",
            "SIM_DUAL_CRASH":90,
            "SIM_DUAL_PHASEONE_METHOD":0,
            "SIM_DUAL_RESTRICT_SELECTION":50,
            "SIM_DUAL_SELECTION":"SIM_SELECTION_FREE",
            "SIM_EXPLOIT_DUPVEC":"SIM_EXPLOIT_DUPVEC_OFF",
            "SIM_HOTSTART":"SIM_HOTSTART_FREE",
            "SIM_HOTSTART_LU":"ON",
            "SIM_INTEGER":0,
            "SIM_MAX_ITERATIONS":10000000,
            "SIM_MAX_NUM_SETBACKS":250,
            "SIM_NON_SINGULAR":"ON",
            "SIM_PRIMAL_CRASH":90,
            "SIM_PRIMAL_PHASEONE_METHOD":0,
            "SIM_PRIMAL_RESTRICT_SELECTION":50,
            "SIM_PRIMAL_SELECTION":"SIM_SELECTION_FREE",
            "SIM_REFACTOR_FREQ":0,
            "SIM_REFORMULATION":"SIM_REFORMULATION_OFF",
            "SIM_SAVE_LU":"OFF",
            "SIM_SCALING":"SCALING_FREE",
            "SIM_SCALING_METHOD":"SCALING_METHOD_POW2",
            "SIM_SOLVE_FORM":"SOLVE_FREE",
            "SIM_STABILITY_PRIORITY":50,
            "SIM_SWITCH_OPTIMIZER":"OFF",
            "SOL_FILTER_KEEP_BASIC":"OFF",
            "SOL_FILTER_KEEP_RANGED":"OFF",
            "SOL_READ_NAME_WIDTH":-1,
            "SOL_READ_WIDTH":1024,
            "SOLUTION_CALLBACK":"OFF",
            "TIMING_LEVEL":1,
            "WRITE_BAS_CONSTRAINTS":"ON",
            "WRITE_BAS_HEAD":"ON",
            "WRITE_BAS_VARIABLES":"ON",
            "WRITE_DATA_COMPRESSED":0,
            "WRITE_DATA_FORMAT":"DATA_FORMAT_EXTENSION",
            "WRITE_DATA_PARAM":"OFF",
            "WRITE_FREE_CON":"OFF",
            "WRITE_GENERIC_NAMES":"OFF",
            "WRITE_GENERIC_NAMES_IO":1,
            "WRITE_IGNORE_INCOMPATIBLE_CONIC_ITEMS":"OFF",
            "WRITE_IGNORE_INCOMPATIBLE_ITEMS":"OFF",
            "WRITE_IGNORE_INCOMPATIBLE_NL_ITEMS":"OFF",
            "WRITE_IGNORE_INCOMPATIBLE_PSD_ITEMS":"OFF",
            "WRITE_INT_CONSTRAINTS":"ON",
            "WRITE_INT_HEAD":"ON",
            "WRITE_INT_VARIABLES":"ON",
            "WRITE_LP_FULL_OBJ":"ON",
            "WRITE_LP_LINE_WIDTH":80,
            "WRITE_LP_QUOTED_NAMES":"ON",
            "WRITE_LP_STRICT_FORMAT":"OFF",
            "WRITE_LP_TERMS_PER_LINE":10,
            "WRITE_MPS_FORMAT":"MPS_FORMAT_FREE",
            "WRITE_MPS_INT":"ON",
            "WRITE_PRECISION":15,
            "WRITE_SOL_BARVARIABLES":"ON",
            "WRITE_SOL_CONSTRAINTS":"ON",
            "WRITE_SOL_HEAD":"ON",
            "WRITE_SOL_IGNORE_INVALID_NAMES":"OFF",
            "WRITE_SOL_VARIABLES":"ON",
```

```
        "WRITE_TASK_INC_SOL":"ON",
        "WRITE_XML_MODE":"WRITE_XML_MODE_ROW"
    },
    "dparam":{
        "ANA_SOL_INFEAS_TOL":1e-6,
        "BASIS_REL_TOL_S":1e-12,
        "BASIS_TOL_S":1e-6,
        "BASIS_TOL_X":1e-6,
        "CHECK_CONVEXITY_REL_TOL":1e-10,
        "DATA_TOL_AIJ":1e-12,
        "DATA_TOL_AIJ_HUGE":1e+20,
        "DATA_TOL_AIJ_LARGE":1e+10,
        "DATA_TOL_BOUND_INF":1e+16,
        "DATA_TOL_BOUND_WRN":1e+8,
        "DATA_TOL_C_HUGE":1e+16,
        "DATA_TOL_CJ_LARGE":1e+8,
        "DATA_TOL_QIJ":1e-16,
        "DATA_TOL_X":1e-8,
        "FEASREPAIR_TOL":1e-10,
        "INTPNT_CO_TOL_DFEAS":1e-8,
        "INTPNT_CO_TOL_INFEAS":1e-10,
        "INTPNT_CO_TOL_MU_RED":1e-8,
        "INTPNT_CO_TOL_NEAR_REL":1e+3,
        "INTPNT_CO_TOL_PFEAS":1e-8,
        "INTPNT_CO_TOL_REL_GAP":1e-7,
        "INTPNT_NL_MERIT_BAL":1e-4,
        "INTPNT_NL_TOL_DFEAS":1e-8,
        "INTPNT_NL_TOL_MU_RED":1e-12,
        "INTPNT_NL_TOL_NEAR_REL":1e+3,
        "INTPNT_NL_TOL_PFEAS":1e-8,
        "INTPNT_NL_TOL_REL_GAP":1e-6,
        "INTPNT_NL_TOL_REL_STEP":9.95e-1,
        "INTPNT_QO_TOL_DFEAS":1e-8,
        "INTPNT_QO_TOL_INFEAS":1e-10,
        "INTPNT_QO_TOL_MU_RED":1e-8,
        "INTPNT_QO_TOL_NEAR_REL":1e+3,
        "INTPNT_QO_TOL_PFEAS":1e-8,
        "INTPNT_QO_TOL_REL_GAP":1e-8,
        "INTPNT_TOL_DFEAS":1e-8,
        "INTPNT_TOL_DSAFE":1e+0,
        "INTPNT_TOL_INFEAS":1e-10,
        "INTPNT_TOL_MU_RED":1e-16,
        "INTPNT_TOL_PATH":1e-8,
        "INTPNT_TOL_PFEAS":1e-8,
        "INTPNT_TOL_PSAFE":1e+0,
        "INTPNT_TOL_REL_GAP":1e-8,
        "INTPNT_TOL_REL_STEP":9.999e-1,
        "INTPNT_TOL_STEP_SIZE":1e-6,
        "LOWER_OBJ_CUT":-1e+30,
        "LOWER_OBJ_CUT_FINITE_TRH":-5e+29,
        "MIO_DISABLE_TERM_TIME":-1e+0,
        "MIO_MAX_TIME":-1e+0,
        "MIO_MAX_TIME_APRX_OPT":6e+1,
        "MIO_NEAR_TOL_ABS_GAP":0.0,
        "MIO_NEAR_TOL_REL_GAP":1e-3,
        "MIO_REL_GAP_CONST":1e-10,
        "MIO_TOL_ABS_GAP":0.0,
        "MIO_TOL_ABS_RELAX_INT":1e-5,
        "MIO_TOL_FEAS":1e-6,
        "MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT":0.0,
        "MIO_TOL_REL_GAP":1e-4,
        "MIO_TOL_X":1e-6,
        "OPTIMIZER_MAX_TIME":-1e+0,
```

```
            "PRESOLVE_TOL_ABS_LINDEP":1e-6,
            "PRESOLVE_TOL_AIJ":1e-12,
            "PRESOLVE_TOL_REL_LINDEP":1e-10,
            "PRESOLVE_TOL_S":1e-8,
            "PRESOLVE_TOL_X":1e-8,
            "QCQO_REFORMULATE_REL_DROP_TOL":1e-15,
            "SEMIDEFINITE_TOL_APPROX":1e-10,
            "SIM_LU_TOL_REL_PIV":1e-2,
            "SIMPLEX_ABS_TOL_PIV":1e-7,
            "UPPER_OBJ_CUT":1e+30,
            "UPPER_OBJ_CUT_FINITE_TRH":5e+29
        },
        "sparam":{
            "BAS_SOL_FILE_NAME":"",
            "DATA_FILE_NAME":"examples/tools/data/lo1.mps",
            "DEBUG_FILE_NAME":"",
            "INT_SOL_FILE_NAME":"",
            "ITR_SOL_FILE_NAME":"",
            "MIO_DEBUG_STRING":"",
            "PARAM_COMMENT_SIGN":"%%",
            "PARAM_READ_FILE_NAME":"",
            "PARAM_WRITE_FILE_NAME":"",
            "READ_MPS_BOU_NAME":"",
            "READ_MPS_OBJ_NAME":"",
            "READ_MPS_RAN_NAME":"",
            "READ_MPS_RHS_NAME":"",
            "SENSITIVITY_FILE_NAME":"",
            "SENSITIVITY_RES_FILE_NAME":"",
            "SOL_FILTER_XC_LOW":"",
            "SOL_FILTER_XC_UPR":"",
            "SOL_FILTER_XX_LOW":"",
            "SOL_FILTER_XX_UPR":"",
            "STAT_FILE_NAME":"",
            "STAT_KEY":"",
            "STAT_NAME":"",
            "WRITE_LP_GEN_VAR_NAME":"XMSKGEN"
        }
    }
}
```

## 18.8 The Solution File Format

**MOSEK** provides several solution files depending on the problem type and the optimizer used:

- *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 contains integer constrained variables.

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>
VARIABLES
INDEX   NAME     AT ACTIVITY    LOWER LIMIT   UPPER LIMIT   DUAL LOWER   DUAL UPPER    CONIC␣
↪DUAL
?       <name>   ?? <a value>   <a value>     <a value>     <a value>    <a value>     <a value>
```

In the example the fields ? and <> will be filled with problem and solution specific information. As can be observed a solution report consists of three sections, i.e.

- HEADER In this section, first the name of the problem is listed and afterwards the problem and solution status are shown. Next the primal and dual objective values are displayed.
- CONSTRAINTS For each constraint $i$ of the form

$$l_i^c \leq \sum_{j=1}^{n} a_{ij} x_j \leq u_i^c, \tag{18.10}$$

  the following information is listed:

  – INDEX: A sequential index assigned to the constraint by **MOSEK**

  – NAME: The name of the constraint assigned by the user.

  – AT: The status of the constraint. In Table 18.4 the possible values of the status keys and their interpretation are shown.

Table 18.4: 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. |

  – ACTIVITY: the quantity $\sum_{j=1}^{n} a_{ij} x_j^*$ , where $x^*$ is the value of the primal solution.

  – LOWER LIMIT: the quantity $l_i^c$ (see (18.10).)

  – UPPER LIMIT: the quantity $u_i^c$ (see (18.10).)

  – DUAL LOWER: the dual multiplier corresponding to the lower limit on the constraint.

  – DUAL UPPER: the dual multiplier corresponding to the upper limit on the constraint.

- VARIABLES The last section of the solution report lists information about the variables. This information has a similar interpretation as for the constraints. However, the column with the header CONIC DUAL is included for problems having one or more conic constraints. This column shows the dual variables corresponding to the conic constraints.

### Example: lo1.sol

In Listing 18.7 we show the solution file for the lo1.opf problem.

Listing 18.7: An example of .sol file.

```
NAME                  :
PROBLEM STATUS        : PRIMAL_AND_DUAL_FEASIBLE
SOLUTION STATUS       : OPTIMAL
OBJECTIVE NAME        : obj
```

```
PRIMAL OBJECTIVE     : 8.33333333e+01
DUAL OBJECTIVE       : 8.33333332e+01

CONSTRAINTS
INDEX      NAME            AT ACTIVITY                 LOWER LIMIT        UPPER LIMIT        ␣
↪DUAL LOWER                 DUAL UPPER
0          c1              EQ 3.00000000000000e+01     3.00000000e+01     3.00000000e+01     -0.
↪00000000000000e+00    -2.49999999741654e+00
1          c2              SB 5.33333333049188e+01     1.50000000e+01     NONE               2.
↪09157603759397e-10     -0.00000000000000e+00
2          c3              UL 2.49999999842049e+01     NONE               2.50000000e+01     -0.
↪00000000000000e+00    -3.33333332895110e-01

VARIABLES
INDEX      NAME            AT ACTIVITY                 LOWER LIMIT        UPPER LIMIT        ␣
↪DUAL LOWER                 DUAL UPPER
0          x1              LL 1.67020427073508e-09     0.00000000e+00     NONE               -4.
↪49999999528055e+00    -0.00000000000000e+00
1          x2              LL 2.93510446280504e-09     0.00000000e+00     1.00000000e+01     -2.
↪16666666494916e+00    6.20863861687316e-10
2          x3              SB 1.49999999899425e+01     0.00000000e+00     NONE               -8.
↪79123177454657e-10    -0.00000000000000e+00
3          x4              SB 8.33333332273116e+00     0.00000000e+00     NONE               -1.
↪69795978899185e-09    -0.00000000000000e+00
```

CHAPTER

# NINETEEN

# LIST OF EXAMPLES

List of examples shipped in the distribution of Optimization Toolbox for MATLAB:

Table 19.1: List of distributed examples

| File | Description |
|---|---|
| `callbackex.m` | An example of a callback function writing to a log file |
| `cqo1.m` | A simple conic quadratic problem |
| `eo1.m` | A simple entropy optimization problem |
| `feasrepairex1.m` | A simple example of how to repair an infeasible problem |
| `go1.m` | A simple geometric optimization problem |
| `go2.m` | A simple geometric optimization problem |
| `lo1.m` | A simple linear problem |
| `lo2.m` | A simple linear problem |
| `milo1.m` | A simple mixed-integer linear problem |
| `mioinitsol.m` | A simple mixed-integer linear problem with an initial guess |
| `nrm1.m` | Solve a linear least-squares problem as a quadratic problem |
| `nrm2.m` | Solve a linear least-squares problem with constraints |
| `nrm3.m` | Solve a linear regression problem with infinity norm |
| `nrm4.m` | Solve a linear regression problem with L1 norm |
| `nrm5.m` | Solve a linear least-squares problem as a reformulated quadratic problem |
| `production.m` | Demonstrate how to modify and re-optimize a linear problem |
| `qcqo1.m` | A simple quadratically constrained quadratic problem |
| `qo1.m` | A simple quadratic problem |
| `qo2.m` | A simple quadratic problem |
| `response.m` | Demonstrates proper response handling |
| `rlo1.m` | Robust linear optimization example, part 1 |
| `rlo2.m` | Robust linear optimization example, part 2 |
| `scopt1.m` | Shows how to solve a simple non-linear separable problem using the SCopt interface |
| `sdo1.m` | A simple semidefinite optimization problem |
| `sensitivity.m` | Sensitivity analysis performed on a small linear problem |
| `sensitivity2.m` | Sensitivity analysis performed on a small linear problem |
| `simple.m` | A simple I/O example: read problem from a file, solve and write solutions |
| `solutionquality.m` | Demonstrates how to examine the quality of a solution |

Additional examples can be found on the **MOSEK** website and in other **MOSEK** publications.

CHAPTER

TWENTY

# INTERFACE CHANGES

The section show interface-specific changes to the **MOSEK** Optimization Toolbox for MATLAB in version 8. See the release notes for general changes and new features of the **MOSEK** Optimization Suite.

## 20.1 Compatibility

- The MATLAB compatibility function `bintprog` has been replaced by *intlinprog* to comform with MATLAB 2014 and later.

Compatibility guarantees for this interface has been updated. See the new *list of supported MATLAB versions*.

## 20.2 Parameters

### Added

- *MSK_DPAR_DATA_SYM_MAT_TOL*
- *MSK_DPAR_DATA_SYM_MAT_TOL_HUGE*
- *MSK_DPAR_DATA_SYM_MAT_TOL_LARGE*
- *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_SEMIDEFINITE_TOL_APPROX*
- *MSK_IPAR_INTPNT_MULTI_THREAD*
- *MSK_IPAR_LICENSE_TRH_EXPIRY_WRN*
- *MSK_IPAR_LOG_ANA_PRO*
- *MSK_IPAR_MIO_CUT_CLIQUE*
- *MSK_IPAR_MIO_CUT_GMI*
- *MSK_IPAR_MIO_CUT_IMPLIED_BOUND*
- *MSK_IPAR_MIO_CUT_KNAPSACK_COVER*

- *MSK_IPAR_MIO_CUT_SELECTION_LEVEL*
- *MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE*
- *MSK_IPAR_MIO_ROOT_REPEAT_PRESOLVE_LEVEL*
- *MSK_IPAR_MIO_VB_DETECTION_LEVEL*
- *MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_FILL*
- *MSK_IPAR_REMOVE_UNUSED_SOLUTIONS*
- *MSK_IPAR_WRITE_LP_FULL_OBJ*
- *MSK_IPAR_WRITE_MPS_FORMAT*
- *MSK_SPAR_REMOTE_ACCESS_TOKEN*

### Removed

- MSK_DPAR_FEASREPAIR_TOL
- MSK_DPAR_MIO_HEURISTIC_TIME
- MSK_DPAR_MIO_MAX_TIME_APRX_OPT
- MSK_DPAR_MIO_REL_ADD_CUT_LIMITED
- MSK_DPAR_MIO_TOL_MAX_CUT_FRAC_RHS
- MSK_DPAR_MIO_TOL_MIN_CUT_FRAC_RHS
- MSK_DPAR_MIO_TOL_REL_RELAX_INT
- MSK_DPAR_MIO_TOL_X
- MSK_DPAR_NONCONVEX_TOL_FEAS
- MSK_DPAR_NONCONVEX_TOL_OPT
- MSK_IPAR_ALLOC_ADD_QNZ
- MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS
- MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX
- MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX
- MSK_IPAR_CONCURRENT_PRIORITY_INTPNT
- MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX
- MSK_IPAR_FEASREPAIR_OPTIMIZE
- MSK_IPAR_INTPNT_FACTOR_DEBUG_LVL
- MSK_IPAR_INTPNT_FACTOR_METHOD
- MSK_IPAR_LIC_TRH_EXPIRY_WRN
- MSK_IPAR_LOG_CONCURRENT
- MSK_IPAR_LOG_FACTOR
- MSK_IPAR_LOG_HEAD
- MSK_IPAR_LOG_NONCONVEX
- MSK_IPAR_LOG_OPTIMIZER
- MSK_IPAR_LOG_PARAM
- MSK_IPAR_LOG_SIM_NETWORK_FREQ
- MSK_IPAR_MIO_BRANCH_PRIORITIES_USE

- MSK_IPAR_MIO_CONT_SOL
- MSK_IPAR_MIO_CUT_CG
- MSK_IPAR_MIO_CUT_LEVEL_ROOT
- MSK_IPAR_MIO_CUT_LEVEL_TREE
- MSK_IPAR_MIO_FEASPUMP_LEVEL
- MSK_IPAR_MIO_HOTSTART
- MSK_IPAR_MIO_KEEP_BASIS
- MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER
- MSK_IPAR_MIO_OPTIMIZER_MODE
- MSK_IPAR_MIO_PRESOLVE_AGGREGATE
- MSK_IPAR_MIO_PRESOLVE_PROBING
- MSK_IPAR_MIO_PRESOLVE_USE
- MSK_IPAR_MIO_STRONG_BRANCH
- MSK_IPAR_MIO_USE_MULTITHREADED_OPTIMIZER
- MSK_IPAR_NONCONVEX_MAX_ITERATIONS
- MSK_IPAR_PRESOLVE_ELIM_FILL
- MSK_IPAR_PRESOLVE_ELIMINATOR_USE
- MSK_IPAR_QO_SEPARABLE_REFORMULATION
- MSK_IPAR_READ_ANZ
- MSK_IPAR_READ_CON
- MSK_IPAR_READ_CONE
- MSK_IPAR_READ_MPS_KEEP_INT
- MSK_IPAR_READ_MPS_OBJ_SENSE
- MSK_IPAR_READ_MPS_RELAX
- MSK_IPAR_READ_QNZ
- MSK_IPAR_READ_VAR
- MSK_IPAR_SIM_INTEGER
- MSK_IPAR_WARNING_LEVEL
- MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_CONIC_ITEMS
- MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_NL_ITEMS
- MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_PSD_ITEMS
- MSK_SPAR_FEASREPAIR_NAME_PREFIX
- MSK_SPAR_FEASREPAIR_NAME_SEPARATOR
- MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL

## 20.3 Constants

### Added

- *"MSK_BRANCH_DIR_FAR"*
- *"MSK_BRANCH_DIR_GUIDED"*
- *"MSK_BRANCH_DIR_NEAR"*
- *"MSK_BRANCH_DIR_PSEUDOCOST"*
- *"MSK_BRANCH_DIR_ROOT_LP"*
- *"MSK_CALLBACK_BEGIN_ROOT_CUTGEN"*
- *"MSK_CALLBACK_BEGIN_TO_CONIC"*
- *"MSK_CALLBACK_END_ROOT_CUTGEN"*
- *"MSK_CALLBACK_END_TO_CONIC"*
- *"MSK_CALLBACK_IM_ROOT_CUTGEN"*
- *"MSK_CALLBACK_SOLVING_REMOTE"*
- *"MSK_DATA_FORMAT_JSON_TASK"*
- *"MSK_DINF_MIO_CLIQUE_SEPARATION_TIME"*
- *"MSK_DINF_MIO_CMIR_SEPARATION_TIME"*
- *"MSK_DINF_MIO_GMI_SEPARATION_TIME"*
- *"MSK_DINF_MIO_IMPLIED_BOUND_TIME"*
- *"MSK_DINF_MIO_KNAPSACK_COVER_SEPARATION_TIME"*
- *"MSK_DINF_QCQO_REFORMULATE_MAX_PERTURBATION"*
- *"MSK_DINF_QCQO_REFORMULATE_WORST_CHOLESKY_COLUMN_SCALING"*
- *"MSK_DINF_QCQO_REFORMULATE_WORST_CHOLESKY_DIAG_SCALING"*
- *"MSK_DINF_SOL_BAS_NRM_BARX"*
- *"MSK_DINF_SOL_BAS_NRM_SLC"*
- *"MSK_DINF_SOL_BAS_NRM_SLX"*
- *"MSK_DINF_SOL_BAS_NRM_SUC"*
- *"MSK_DINF_SOL_BAS_NRM_SUX"*
- *"MSK_DINF_SOL_BAS_NRM_XC"*
- *"MSK_DINF_SOL_BAS_NRM_XX"*
- *"MSK_DINF_SOL_BAS_NRM_Y"*
- *"MSK_DINF_SOL_ITG_NRM_BARX"*
- *"MSK_DINF_SOL_ITG_NRM_XC"*
- *"MSK_DINF_SOL_ITG_NRM_XX"*
- *"MSK_DINF_SOL_ITR_NRM_BARS"*
- *"MSK_DINF_SOL_ITR_NRM_BARX"*
- *"MSK_DINF_SOL_ITR_NRM_SLC"*
- *"MSK_DINF_SOL_ITR_NRM_SLX"*

- *"MSK_DINF_SOL_ITR_NRM_SNX"*
- *"MSK_DINF_SOL_ITR_NRM_SUC"*
- *"MSK_DINF_SOL_ITR_NRM_SUX"*
- *"MSK_DINF_SOL_ITR_NRM_XC"*
- *"MSK_DINF_SOL_ITR_NRM_XX"*
- *"MSK_DINF_SOL_ITR_NRM_Y"*
- *"MSK_DINF_TO_CONIC_TIME"*
- *"MSK_IINF_MIO_ABSGAP_SATISFIED"*
- *"MSK_IINF_MIO_CLIQUE_TABLE_SIZE"*
- *"MSK_IINF_MIO_NEAR_ABSGAP_SATISFIED"*
- *"MSK_IINF_MIO_NEAR_RELGAP_SATISFIED"*
- *"MSK_IINF_MIO_NODE_DEPTH"*
- *"MSK_IINF_MIO_NUM_CMIR_CUTS"*
- *"MSK_IINF_MIO_NUM_IMPLIED_BOUND_CUTS"*
- *"MSK_IINF_MIO_NUM_KNAPSACK_COVER_CUTS"*
- *"MSK_IINF_MIO_NUM_REPEATED_PRESOLVE"*
- *"MSK_IINF_MIO_PRESOLVED_NUMBIN"*
- *"MSK_IINF_MIO_PRESOLVED_NUMCON"*
- *"MSK_IINF_MIO_PRESOLVED_NUMCONT"*
- *"MSK_IINF_MIO_PRESOLVED_NUMINT"*
- *"MSK_IINF_MIO_PRESOLVED_NUMVAR"*
- *"MSK_IINF_MIO_RELGAP_SATISFIED"*
- *"MSK_LIINF_MIO_PRESOLVED_ANZ"*
- *"MSK_LIINF_MIO_SIM_MAXITER_SETBACKS"*
- *"MSK_MPS_FORMAT_CPLEX"*
- *"MSK_SOL_STA_DUAL_ILLPOSED_CER"*
- *"MSK_SOL_STA_PRIM_ILLPOSED_CER"*

### Changed

- *"MSK_SOL_STA_INTEGER_OPTIMAL"*
- *"MSK_SOL_STA_NEAR_DUAL_FEAS"*
- *"MSK_SOL_STA_NEAR_DUAL_INFEAS_CER"*
- *"MSK_SOL_STA_NEAR_INTEGER_OPTIMAL"*
- *"MSK_SOL_STA_NEAR_OPTIMAL"*
- *"MSK_SOL_STA_NEAR_PRIM_AND_DUAL_FEAS"*
- *"MSK_SOL_STA_NEAR_PRIM_FEAS"*
- *"MSK_SOL_STA_NEAR_PRIM_INFEAS_CER"*
- *"MSK_LICENSE_BUFFER_LENGTH"*

### Removed

- MSK_CALLBACKCODE_BEGIN_CONCURRENT
- MSK_CALLBACKCODE_BEGIN_NETWORK_DUAL_SIMPLEX
- MSK_CALLBACKCODE_BEGIN_NETWORK_PRIMAL_SIMPLEX
- MSK_CALLBACKCODE_BEGIN_NETWORK_SIMPLEX
- MSK_CALLBACKCODE_BEGIN_NONCONVEX
- MSK_CALLBACKCODE_BEGIN_PRIMAL_DUAL_SIMPLEX
- MSK_CALLBACKCODE_BEGIN_PRIMAL_DUAL_SIMPLEX_BI
- MSK_CALLBACKCODE_BEGIN_SIMPLEX_NETWORK_DETECT
- MSK_CALLBACKCODE_END_CONCURRENT
- MSK_CALLBACKCODE_END_NETWORK_DUAL_SIMPLEX
- MSK_CALLBACKCODE_END_NETWORK_PRIMAL_SIMPLEX
- MSK_CALLBACKCODE_END_NETWORK_SIMPLEX
- MSK_CALLBACKCODE_END_NONCONVEX
- MSK_CALLBACKCODE_END_PRIMAL_DUAL_SIMPLEX
- MSK_CALLBACKCODE_END_PRIMAL_DUAL_SIMPLEX_BI
- MSK_CALLBACKCODE_END_SIMPLEX_NETWORK_DETECT
- MSK_CALLBACKCODE_IM_MIO_PRESOLVE
- MSK_CALLBACKCODE_IM_NETWORK_DUAL_SIMPLEX
- MSK_CALLBACKCODE_IM_NETWORK_PRIMAL_SIMPLEX
- MSK_CALLBACKCODE_IM_NONCONVEX
- MSK_CALLBACKCODE_IM_PRIMAL_DUAL_SIMPLEX
- MSK_CALLBACKCODE_NONCOVEX
- MSK_CALLBACKCODE_UPDATE_NETWORK_DUAL_SIMPLEX
- MSK_CALLBACKCODE_UPDATE_NETWORK_PRIMAL_SIMPLEX
- MSK_CALLBACKCODE_UPDATE_NONCONVEX
- MSK_CALLBACKCODE_UPDATE_PRIMAL_DUAL_SIMPLEX
- MSK_CALLBACKCODE_UPDATE_PRIMAL_DUAL_SIMPLEX_BI
- MSK_DINFITEM_BI_CLEAN_PRIMAL_DUAL_TIME
- MSK_DINFITEM_CONCURRENT_TIME
- MSK_DINFITEM_MIO_CG_SEPERATION_TIME
- MSK_DINFITEM_MIO_CMIR_SEPERATION_TIME
- MSK_DINFITEM_SIM_NETWORK_DUAL_TIME
- MSK_DINFITEM_SIM_NETWORK_PRIMAL_TIME
- MSK_DINFITEM_SIM_NETWORK_TIME
- MSK_DINFITEM_SIM_PRIMAL_DUAL_TIME
- MSK_FEATURE_PTOM
- MSK_FEATURE_PTOX

- MSK_IINFITEM_CONCURRENT_FASTEST_OPTIMIZER
- MSK_IINFITEM_MIO_NUM_BASIS_CUTS
- MSK_IINFITEM_MIO_NUM_CARDGUB_CUTS
- MSK_IINFITEM_MIO_NUM_COEF_REDC_CUTS
- MSK_IINFITEM_MIO_NUM_CONTRA_CUTS
- MSK_IINFITEM_MIO_NUM_DISAGG_CUTS
- MSK_IINFITEM_MIO_NUM_FLOW_COVER_CUTS
- MSK_IINFITEM_MIO_NUM_GCD_CUTS
- MSK_IINFITEM_MIO_NUM_GUB_COVER_CUTS
- MSK_IINFITEM_MIO_NUM_KNAPSUR_COVER_CUTS
- MSK_IINFITEM_MIO_NUM_LATTICE_CUTS
- MSK_IINFITEM_MIO_NUM_LIFT_CUTS
- MSK_IINFITEM_MIO_NUM_OBJ_CUTS
- MSK_IINFITEM_MIO_NUM_PLAN_LOC_CUTS
- MSK_IINFITEM_SIM_NETWORK_DUAL_DEG_ITER
- MSK_IINFITEM_SIM_NETWORK_DUAL_HOTSTART
- MSK_IINFITEM_SIM_NETWORK_DUAL_HOTSTART_LU
- MSK_IINFITEM_SIM_NETWORK_DUAL_INF_ITER
- MSK_IINFITEM_SIM_NETWORK_DUAL_ITER
- MSK_IINFITEM_SIM_NETWORK_PRIMAL_DEG_ITER
- MSK_IINFITEM_SIM_NETWORK_PRIMAL_HOTSTART
- MSK_IINFITEM_SIM_NETWORK_PRIMAL_HOTSTART_LU
- MSK_IINFITEM_SIM_NETWORK_PRIMAL_INF_ITER
- MSK_IINFITEM_SIM_NETWORK_PRIMAL_ITER
- MSK_IINFITEM_SIM_PRIMAL_DUAL_DEG_ITER
- MSK_IINFITEM_SIM_PRIMAL_DUAL_HOTSTART
- MSK_IINFITEM_SIM_PRIMAL_DUAL_HOTSTART_LU
- MSK_IINFITEM_SIM_PRIMAL_DUAL_INF_ITER
- MSK_IINFITEM_SIM_PRIMAL_DUAL_ITER
- MSK_IINFITEM_SOL_INT_PROSTA
- MSK_IINFITEM_SOL_INT_SOLSTA
- MSK_IINFITEM_STO_NUM_A_CACHE_FLUSHES
- MSK_IINFITEM_STO_NUM_A_TRANSPOSES
- MSK_LIINFITEM_BI_CLEAN_PRIMAL_DUAL_DEG_ITER
- MSK_LIINFITEM_BI_CLEAN_PRIMAL_DUAL_ITER
- MSK_LIINFITEM_BI_CLEAN_PRIMAL_DUAL_SUB_ITER
- MSK_MIOMODE_LAZY
- MSK_OPTIMIZERTYPE_CONCURRENT
- MSK_OPTIMIZERTYPE_MIXED_INT_CONIC

- MSK_OPTIMIZERTYPE_NETWORK_PRIMAL_SIMPLEX
- MSK_OPTIMIZERTYPE_NONCONVEX
- MSK_OPTIMIZERTYPE_PRIMAL_DUAL_SIMPLEX

## 20.4 Response Codes

### Added

- *"MSK_RES_ERR_CBF_DUPLICATE_PSDVAR"*
- *"MSK_RES_ERR_CBF_INVALID_PSDVAR_DIMENSION"*
- *"MSK_RES_ERR_CBF_TOO_FEW_PSDVAR"*
- *"MSK_RES_ERR_DUPLICATE_AIJ"*
- *"MSK_RES_ERR_FINAL_SOLUTION"*
- *"MSK_RES_ERR_JSON_DATA"*
- *"MSK_RES_ERR_JSON_FORMAT"*
- *"MSK_RES_ERR_JSON_MISSING_DATA"*
- *"MSK_RES_ERR_JSON_NUMBER_OVERFLOW"*
- *"MSK_RES_ERR_JSON_STRING"*
- *"MSK_RES_ERR_JSON_SYNTAX"*
- *"MSK_RES_ERR_LAU_INVALID_LOWER_TRIANGULAR_MATRIX"*
- *"MSK_RES_ERR_LAU_INVALID_SPARSE_SYMMETRIC_MATRIX"*
- *"MSK_RES_ERR_LAU_NOT_POSITIVE_DEFINITE"*
- *"MSK_RES_ERR_MIXED_CONIC_AND_NL"*
- *"MSK_RES_ERR_SERVER_CONNECT"*
- *"MSK_RES_ERR_SERVER_PROTOCOL"*
- *"MSK_RES_ERR_SERVER_STATUS"*
- *"MSK_RES_ERR_SERVER_TOKEN"*
- *"MSK_RES_ERR_SYM_MAT_HUGE"*
- *"MSK_RES_ERR_SYM_MAT_INVALID"*
- *"MSK_RES_ERR_TASK_WRITE"*
- *"MSK_RES_ERR_TOCONIC_CONSTR_NOT_CONIC"*
- *"MSK_RES_ERR_TOCONIC_CONSTR_Q_NOT_PSD"*
- *"MSK_RES_ERR_TOCONIC_CONSTRAINT_FX"*
- *"MSK_RES_ERR_TOCONIC_CONSTRAINT_RA"*
- *"MSK_RES_ERR_TOCONIC_OBJECTIVE_NOT_PSD"*
- *"MSK_RES_WRN_SYM_MAT_LARGE"*

### Removed

- MSK_RES_ERR_AD_INVALID_OPERAND
- MSK_RES_ERR_AD_INVALID_OPERATOR
- MSK_RES_ERR_AD_MISSING_OPERAND
- MSK_RES_ERR_AD_MISSING_RETURN
- MSK_RES_ERR_CONCURRENT_OPTIMIZER
- MSK_RES_ERR_INV_CONIC_PROBLEM
- MSK_RES_ERR_INVALID_BRANCH_DIRECTION
- MSK_RES_ERR_INVALID_BRANCH_PRIORITY
- MSK_RES_ERR_INVALID_NETWORK_PROBLEM
- MSK_RES_ERR_MBT_INCOMPATIBLE
- MSK_RES_ERR_MBT_INVALID
- MSK_RES_ERR_MIO_NOT_LOADED
- MSK_RES_ERR_MIXED_PROBLEM
- MSK_RES_ERR_NO_DUAL_INFO_FOR_ITG_SOL
- MSK_RES_ERR_ORD_INVALID
- MSK_RES_ERR_ORD_INVALID_BRANCH_DIR
- MSK_RES_ERR_TOCONIC_CONVERSION_FAIL
- MSK_RES_ERR_TOO_MANY_CONCURRENT_TASKS
- MSK_RES_WRN_TOO_MANY_THREADS_CONCURRENT

# BIBLIOGRAPHY

[AA95] E. D. Andersen and K. D. Andersen. Presolving in linear programming. *Math. Programming*, 71(2):221–245, 1995.

[AGMX96] 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.

[AY98] E. D. Andersen and Y. Ye. A computational study of the homogeneous algorithm for large-scale convex optimization. *Computational Optimization and Applications*, 10:243–269, 1998.

[AY99] E. D. Andersen and Y. Ye. On a homogeneous algorithm for the monotone complementarity problem. *Math. Programming*, 84(2):375–399, February 1999.

[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.

[BSS93] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming: Theory and algorithms.* John Wiley and Sons, New York, 2 edition, 1993.

[BP76] C. Beightler and D. T. Phillips. *Applied geometric programming.* John Wiley and Sons, New York, 1976.

[BTN00] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Math. Programming*, 88(3):411–424, 2000.

[BKVH04] S. Boyd, S.J. Kim, L. Vandenberghe, and A. Hassibi. A Tutorial on Geometric Programming. Technical Report, ISL, Electrical Engineering Department, Stanford University, Stanford, CA, 2004. Available at http://www.stanford.edu/~boyd/gp_tutorial.html.

[Chv83] V. Chvátal. *Linear programming.* W.H. Freeman and Company, 1983.

[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.

[BenTalN01] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. MPS/SIAM Series on Optimization. SIAM, 2001.

[MOSEKApS12] MOSEK ApS. *The MOSEK Modeling Cookbook*. MOSEK ApS, Fruebjergvej 3, Boks 16, 2100 Copenhagen O, 2012. Last revised September 2015. URL: http://docs.mosek.com/generic/modeling-a4.pdf.

# SYMBOL INDEX

## Enumerations


accmode, 221
"MSK_ACC_VAR", 221
"MSK_ACC_CON", 221
basindtype, 221
"MSK_BI_RESERVERED", 221
"MSK_BI_NO_ERROR", 221
"MSK_BI_NEVER", 221
"MSK_BI_IF_FEASIBLE", 221
"MSK_BI_ALWAYS", 221
boundkey, 221
"MSK_BK_UP", 221
"MSK_BK_RA", 221
"MSK_BK_LO", 221
"MSK_BK_FX", 221
"MSK_BK_FR", 221
branchdir, 238
"MSK_BRANCH_DIR_UP", 238
"MSK_BRANCH_DIR_ROOT_LP", 238
"MSK_BRANCH_DIR_PSEUDOCOST", 238
"MSK_BRANCH_DIR_NEAR", 238
"MSK_BRANCH_DIR_GUIDED", 238
"MSK_BRANCH_DIR_FREE", 238
"MSK_BRANCH_DIR_FAR", 238
"MSK_BRANCH_DIR_DOWN", 238
callbackcode, 223
"MSK_CALLBACK_WRITE_OPF", 227
"MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX_BI", 227
"MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX", 227
"MSK_CALLBACK_UPDATE_PRIMAL_BI", 227
"MSK_CALLBACK_UPDATE_PRESOLVE", 227
"MSK_CALLBACK_UPDATE_DUAL_SIMPLEX_BI", 227
"MSK_CALLBACK_UPDATE_DUAL_SIMPLEX", 227
"MSK_CALLBACK_UPDATE_DUAL_BI", 227
"MSK_CALLBACK_SOLVING_REMOTE", 227
"MSK_CALLBACK_READ_OPF_SECTION", 227
"MSK_CALLBACK_READ_OPF", 227
"MSK_CALLBACK_PRIMAL_SIMPLEX", 227
"MSK_CALLBACK_NEW_INT_MIO", 227
"MSK_CALLBACK_INTPNT", 227
"MSK_CALLBACK_IM_SIMPLEX_BI", 227
"MSK_CALLBACK_IM_SIMPLEX", 227
"MSK_CALLBACK_IM_ROOT_CUTGEN", 227
"MSK_CALLBACK_IM_READ", 226
"MSK_CALLBACK_IM_QO_REFORMULATE", 226
"MSK_CALLBACK_IM_PRIMAL_SIMPLEX", 226
"MSK_CALLBACK_IM_PRIMAL_SENSIVITY", 226
"MSK_CALLBACK_IM_PRIMAL_BI", 226
"MSK_CALLBACK_IM_PRESOLVE", 226
"MSK_CALLBACK_IM_ORDER", 226
"MSK_CALLBACK_IM_MIO_PRIMAL_SIMPLEX", 226
"MSK_CALLBACK_IM_MIO_INTPNT", 226
"MSK_CALLBACK_IM_MIO_DUAL_SIMPLEX", 226
"MSK_CALLBACK_IM_MIO", 226
"MSK_CALLBACK_IM_LU", 226
"MSK_CALLBACK_IM_LICENSE_WAIT", 226
"MSK_CALLBACK_IM_INTPNT", 226
"MSK_CALLBACK_IM_FULL_CONVEXITY_CHECK", 226
"MSK_CALLBACK_IM_DUAL_SIMPLEX", 226
"MSK_CALLBACK_IM_DUAL_SENSIVITY", 226
"MSK_CALLBACK_IM_DUAL_BI", 226
"MSK_CALLBACK_IM_CONIC", 226
"MSK_CALLBACK_IM_BI", 225
"MSK_CALLBACK_END_WRITE", 225
"MSK_CALLBACK_END_TO_CONIC", 225
"MSK_CALLBACK_END_SIMPLEX_BI", 225
"MSK_CALLBACK_END_SIMPLEX", 225
"MSK_CALLBACK_END_ROOT_CUTGEN", 225
"MSK_CALLBACK_END_READ", 225
"MSK_CALLBACK_END_QCQO_REFORMULATE", 225
"MSK_CALLBACK_END_PRIMAL_SIMPLEX_BI", 225
"MSK_CALLBACK_END_PRIMAL_SIMPLEX", 225
"MSK_CALLBACK_END_PRIMAL_SETUP_BI", 225
"MSK_CALLBACK_END_PRIMAL_SENSITIVITY", 225
"MSK_CALLBACK_END_PRIMAL_REPAIR", 225
"MSK_CALLBACK_END_PRIMAL_BI", 225
"MSK_CALLBACK_END_PRESOLVE", 225
"MSK_CALLBACK_END_OPTIMIZER", 225
"MSK_CALLBACK_END_MIO", 225
"MSK_CALLBACK_END_LICENSE_WAIT", 225
"MSK_CALLBACK_END_INTPNT", 225
"MSK_CALLBACK_END_INFEAS_ANA", 225
"MSK_CALLBACK_END_FULL_CONVEXITY_CHECK", 225
"MSK_CALLBACK_END_DUAL_SIMPLEX_BI", 224
"MSK_CALLBACK_END_DUAL_SIMPLEX", 224
"MSK_CALLBACK_END_DUAL_SETUP_BI", 224
"MSK_CALLBACK_END_DUAL_SENSITIVITY", 224
"MSK_CALLBACK_END_DUAL_BI", 224
"MSK_CALLBACK_END_CONIC", 224
"MSK_CALLBACK_END_BI", 224

"MSK_CALLBACK_DUAL_SIMPLEX", 224
"MSK_CALLBACK_CONIC", 224
"MSK_CALLBACK_BEGIN_WRITE", 224
"MSK_CALLBACK_BEGIN_TO_CONIC", 224
"MSK_CALLBACK_BEGIN_SIMPLEX_BI", 224
"MSK_CALLBACK_BEGIN_SIMPLEX", 224
"MSK_CALLBACK_BEGIN_ROOT_CUTGEN", 224
"MSK_CALLBACK_BEGIN_READ", 224
"MSK_CALLBACK_BEGIN_QCQO_REFORMULATE", 224
"MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX_BI", 224
"MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX", 224
"MSK_CALLBACK_BEGIN_PRIMAL_SETUP_BI", 224
"MSK_CALLBACK_BEGIN_PRIMAL_SENSITIVITY", 224
"MSK_CALLBACK_BEGIN_PRIMAL_REPAIR", 224
"MSK_CALLBACK_BEGIN_PRIMAL_BI", 223
"MSK_CALLBACK_BEGIN_PRESOLVE", 223
"MSK_CALLBACK_BEGIN_OPTIMIZER", 223
"MSK_CALLBACK_BEGIN_MIO", 223
"MSK_CALLBACK_BEGIN_LICENSE_WAIT", 223
"MSK_CALLBACK_BEGIN_INTPNT", 223
"MSK_CALLBACK_BEGIN_INFEAS_ANA", 223
"MSK_CALLBACK_BEGIN_FULL_CONVEXITY_CHECK", 223
"MSK_CALLBACK_BEGIN_DUAL_SIMPLEX_BI", 223
"MSK_CALLBACK_BEGIN_DUAL_SIMPLEX", 223
"MSK_CALLBACK_BEGIN_DUAL_SETUP_BI", 223
"MSK_CALLBACK_BEGIN_DUAL_SENSITIVITY", 223
"MSK_CALLBACK_BEGIN_DUAL_BI", 223
"MSK_CALLBACK_BEGIN_CONIC", 223
"MSK_CALLBACK_BEGIN_BI", 223
checkconvexitytype, 227
"MSK_CHECK_CONVEXITY_SIMPLE", 228
"MSK_CHECK_CONVEXITY_NONE", 227
"MSK_CHECK_CONVEXITY_FULL", 228
compresstype, 228
"MSK_COMPRESS_NONE", 228
"MSK_COMPRESS_GZIP", 228
"MSK_COMPRESS_FREE", 228
conetype, 228
"MSK_CT_RQUAD", 228
"MSK_CT_QUAD", 228
dataformat, 228
"MSK_DATA_FORMAT_XML", 228
"MSK_DATA_FORMAT_TASK", 228
"MSK_DATA_FORMAT_OP", 228
"MSK_DATA_FORMAT_MPS", 228
"MSK_DATA_FORMAT_LP", 228
"MSK_DATA_FORMAT_JSON_TASK", 229
"MSK_DATA_FORMAT_FREE_MPS", 228
"MSK_DATA_FORMAT_EXTENSION", 228
"MSK_DATA_FORMAT_CB", 229
dinfitem, 229
"MSK_DINF_TO_CONIC_TIME", 233
"MSK_DINF_SOL_ITR_PVIOLVAR", 233
"MSK_DINF_SOL_ITR_PVIOLCONES", 233
"MSK_DINF_SOL_ITR_PVIOLCON", 233
"MSK_DINF_SOL_ITR_PVIOLBARVAR", 233
"MSK_DINF_SOL_ITR_PRIMAL_OBJ", 233
"MSK_DINF_SOL_ITR_NRM_Y", 233
"MSK_DINF_SOL_ITR_NRM_XX", 233
"MSK_DINF_SOL_ITR_NRM_XC", 233
"MSK_DINF_SOL_ITR_NRM_SUX", 233
"MSK_DINF_SOL_ITR_NRM_SUC", 233
"MSK_DINF_SOL_ITR_NRM_SNX", 233
"MSK_DINF_SOL_ITR_NRM_SLX", 232
"MSK_DINF_SOL_ITR_NRM_SLC", 232
"MSK_DINF_SOL_ITR_NRM_BARX", 232
"MSK_DINF_SOL_ITR_NRM_BARS", 232
"MSK_DINF_SOL_ITR_DVIOLVAR", 232
"MSK_DINF_SOL_ITR_DVIOLCONES", 232
"MSK_DINF_SOL_ITR_DVIOLCON", 232
"MSK_DINF_SOL_ITR_DVIOLBARVAR", 232
"MSK_DINF_SOL_ITR_DUAL_OBJ", 232
"MSK_DINF_SOL_ITG_PVIOLVAR", 232
"MSK_DINF_SOL_ITG_PVIOLITG", 232
"MSK_DINF_SOL_ITG_PVIOLCONES", 232
"MSK_DINF_SOL_ITG_PVIOLCON", 232
"MSK_DINF_SOL_ITG_PVIOLBARVAR", 232
"MSK_DINF_SOL_ITG_PRIMAL_OBJ", 232
"MSK_DINF_SOL_ITG_NRM_XX", 232
"MSK_DINF_SOL_ITG_NRM_XC", 232
"MSK_DINF_SOL_ITG_NRM_BARX", 232
"MSK_DINF_SOL_BAS_PVIOLVAR", 232
"MSK_DINF_SOL_BAS_PVIOLCON", 232
"MSK_DINF_SOL_BAS_PRIMAL_OBJ", 232
"MSK_DINF_SOL_BAS_NRM_Y", 232
"MSK_DINF_SOL_BAS_NRM_XX", 232
"MSK_DINF_SOL_BAS_NRM_XC", 231
"MSK_DINF_SOL_BAS_NRM_SUX", 231
"MSK_DINF_SOL_BAS_NRM_SUC", 231
"MSK_DINF_SOL_BAS_NRM_SLX", 231
"MSK_DINF_SOL_BAS_NRM_SLC", 231
"MSK_DINF_SOL_BAS_NRM_BARX", 231
"MSK_DINF_SOL_BAS_DVIOLVAR", 231
"MSK_DINF_SOL_BAS_DVIOLCON", 231
"MSK_DINF_SOL_BAS_DUAL_OBJ", 231
"MSK_DINF_SIM_TIME", 231
"MSK_DINF_SIM_PRIMAL_TIME", 231
"MSK_DINF_SIM_OBJ", 231
"MSK_DINF_SIM_FEAS", 231
"MSK_DINF_SIM_DUAL_TIME", 231
"MSK_DINF_RD_TIME", 231
"MSK_DINF_QCQO_REFORMULATE_WORST_CHOLESKY_DIAG_SCALING", 231
"MSK_DINF_QCQO_REFORMULATE_WORST_CHOLESKY_COLUMN_SCALING 231
"MSK_DINF_QCQO_REFORMULATE_TIME", 231
"MSK_DINF_QCQO_REFORMULATE_MAX_PERTURBATION", 231
"MSK_DINF_PRIMAL_REPAIR_PENALTY_OBJ", 231
"MSK_DINF_PRESOLVE_TIME", 231
"MSK_DINF_PRESOLVE_LINDEP_TIME", 231
"MSK_DINF_PRESOLVE_ELI_TIME", 231
"MSK_DINF_OPTIMIZER_TIME", 230
"MSK_DINF_MIO_USER_OBJ_CUT", 230

"MSK_DINF_MIO_TIME", 230
"MSK_DINF_MIO_ROOT_PRESOLVE_TIME", 230
"MSK_DINF_MIO_ROOT_OPTIMIZER_TIME", 230
"MSK_DINF_MIO_ROOT_CUTGEN_TIME", 230
"MSK_DINF_MIO_PROBING_TIME", 230
"MSK_DINF_MIO_OPTIMIZER_TIME", 230
"MSK_DINF_MIO_OBJ_REL_GAP", 230
"MSK_DINF_MIO_OBJ_INT", 230
"MSK_DINF_MIO_OBJ_BOUND", 230
"MSK_DINF_MIO_OBJ_ABS_GAP", 230
"MSK_DINF_MIO_KNAPSACK_COVER_SEPARATION_TIME", 230
"MSK_DINF_MIO_IMPLIED_BOUND_TIME", 230
"MSK_DINF_MIO_HEURISTIC_TIME", 230
"MSK_DINF_MIO_GMI_SEPARATION_TIME", 230
"MSK_DINF_MIO_DUAL_BOUND_AFTER_PRESOLVE", 230
"MSK_DINF_MIO_CONSTRUCT_SOLUTION_OBJ", 229
"MSK_DINF_MIO_CMIR_SEPARATION_TIME", 229
"MSK_DINF_MIO_CLIQUE_SEPARATION_TIME", 229
"MSK_DINF_INTPNT_TIME", 229
"MSK_DINF_INTPNT_PRIMAL_OBJ", 229
"MSK_DINF_INTPNT_PRIMAL_FEAS", 229
"MSK_DINF_INTPNT_ORDER_TIME", 229
"MSK_DINF_INTPNT_OPT_STATUS", 229
"MSK_DINF_INTPNT_FACTOR_NUM_FLOPS", 229
"MSK_DINF_INTPNT_DUAL_OBJ", 229
"MSK_DINF_INTPNT_DUAL_FEAS", 229
"MSK_DINF_BI_TIME", 229
"MSK_DINF_BI_PRIMAL_TIME", 229
"MSK_DINF_BI_DUAL_TIME", 229
"MSK_DINF_BI_CLEAN_TIME", 229
"MSK_DINF_BI_CLEAN_PRIMAL_TIME", 229
"MSK_DINF_BI_CLEAN_DUAL_TIME", 229
dparam, 162
feature, 233
"MSK_FEATURE_PTS", 233
"MSK_FEATURE_PTON", 233
iinfitem, 234
"MSK_IINF_STO_NUM_A_REALLOC", 237
"MSK_IINF_SOL_ITR_SOLSTA", 237
"MSK_IINF_SOL_ITR_PROSTA", 237
"MSK_IINF_SOL_ITG_SOLSTA", 237
"MSK_IINF_SOL_ITG_PROSTA", 237
"MSK_IINF_SOL_BAS_SOLSTA", 237
"MSK_IINF_SOL_BAS_PROSTA", 237
"MSK_IINF_SIM_SOLVE_DUAL", 237
"MSK_IINF_SIM_PRIMAL_ITER", 237
"MSK_IINF_SIM_PRIMAL_INF_ITER", 237
"MSK_IINF_SIM_PRIMAL_HOTSTART_LU", 237
"MSK_IINF_SIM_PRIMAL_HOTSTART", 237
"MSK_IINF_SIM_PRIMAL_DEG_ITER", 237
"MSK_IINF_SIM_NUMVAR", 237
"MSK_IINF_SIM_NUMCON", 237
"MSK_IINF_SIM_DUAL_ITER", 237
"MSK_IINF_SIM_DUAL_INF_ITER", 237
"MSK_IINF_SIM_DUAL_HOTSTART_LU", 237
"MSK_IINF_SIM_DUAL_HOTSTART", 236
"MSK_IINF_SIM_DUAL_DEG_ITER", 236
"MSK_IINF_RD_PROTYPE", 236
"MSK_IINF_RD_NUMVAR", 236
"MSK_IINF_RD_NUMQ", 236
"MSK_IINF_RD_NUMINTVAR", 236
"MSK_IINF_RD_NUMCONE", 236
"MSK_IINF_RD_NUMCON", 236
"MSK_IINF_RD_NUMBARVAR", 236
"MSK_IINF_OPTIMIZE_RESPONSE", 236
"MSK_IINF_OPT_NUMVAR", 236
"MSK_IINF_OPT_NUMCON", 236
"MSK_IINF_MIO_USER_OBJ_CUT", 236
"MSK_IINF_MIO_TOTAL_NUM_CUTS", 236
"MSK_IINF_MIO_RELGAP_SATISFIED", 236
"MSK_IINF_MIO_PRESOLVED_NUMVAR", 236
"MSK_IINF_MIO_PRESOLVED_NUMINT", 236
"MSK_IINF_MIO_PRESOLVED_NUMCONT", 236
"MSK_IINF_MIO_PRESOLVED_NUMCON", 236
"MSK_IINF_MIO_PRESOLVED_NUMBIN", 236
"MSK_IINF_MIO_OBJ_BOUND_DEFINED", 236
"MSK_IINF_MIO_NUMVAR", 236
"MSK_IINF_MIO_NUMINT", 236
"MSK_IINF_MIO_NUMCON", 235
"MSK_IINF_MIO_NUM_REPEATED_PRESOLVE", 235
"MSK_IINF_MIO_NUM_RELAX", 235
"MSK_IINF_MIO_NUM_KNAPSACK_COVER_CUTS", 235
"MSK_IINF_MIO_NUM_INT_SOLUTIONS", 235
"MSK_IINF_MIO_NUM_IMPLIED_BOUND_CUTS", 235
"MSK_IINF_MIO_NUM_GOMORY_CUTS", 235
"MSK_IINF_MIO_NUM_CMIR_CUTS", 235
"MSK_IINF_MIO_NUM_CLIQUE_CUTS", 235
"MSK_IINF_MIO_NUM_BRANCH", 235
"MSK_IINF_MIO_NUM_ACTIVE_NODES", 235
"MSK_IINF_MIO_NODE_DEPTH", 235
"MSK_IINF_MIO_NEAR_RELGAP_SATISFIED", 235
"MSK_IINF_MIO_NEAR_ABSGAP_SATISFIED", 235
"MSK_IINF_MIO_INITIAL_SOLUTION", 235
"MSK_IINF_MIO_CONSTRUCT_SOLUTION", 235
"MSK_IINF_MIO_CONSTRUCT_NUM_ROUNDINGS", 235
"MSK_IINF_MIO_CLIQUE_TABLE_SIZE", 235
"MSK_IINF_MIO_ABSGAP_SATISFIED", 235
"MSK_IINF_INTPNT_SOLVE_DUAL", 235
"MSK_IINF_INTPNT_NUM_THREADS", 235
"MSK_IINF_INTPNT_ITER", 235
"MSK_IINF_INTPNT_FACTOR_DIM_DENSE", 234
"MSK_IINF_ANA_PRO_NUM_VAR_UP", 234
"MSK_IINF_ANA_PRO_NUM_VAR_RA", 234
"MSK_IINF_ANA_PRO_NUM_VAR_LO", 234
"MSK_IINF_ANA_PRO_NUM_VAR_INT", 234
"MSK_IINF_ANA_PRO_NUM_VAR_FR", 234
"MSK_IINF_ANA_PRO_NUM_VAR_EQ", 234
"MSK_IINF_ANA_PRO_NUM_VAR_CONT", 234
"MSK_IINF_ANA_PRO_NUM_VAR_BIN", 234
"MSK_IINF_ANA_PRO_NUM_VAR", 234
"MSK_IINF_ANA_PRO_NUM_CON_UP", 234
"MSK_IINF_ANA_PRO_NUM_CON_RA", 234
"MSK_IINF_ANA_PRO_NUM_CON_LO", 234
"MSK_IINF_ANA_PRO_NUM_CON_FR", 234

"MSK_IINF_ANA_PRO_NUM_CON_EQ", 234
"MSK_IINF_ANA_PRO_NUM_CON", 234
inftype, 237
"MSK_INF_LINT_TYPE", 237
"MSK_INF_INT_TYPE", 237
"MSK_INF_DOU_TYPE", 237
intpnthotstart, 223
"MSK_INTPNT_HOTSTART_PRIMAL_DUAL", 223
"MSK_INTPNT_HOTSTART_PRIMAL", 223
"MSK_INTPNT_HOTSTART_NONE", 223
"MSK_INTPNT_HOTSTART_DUAL", 223
iomode, 238
"MSK_IOMODE_WRITE", 238
"MSK_IOMODE_READWRITE", 238
"MSK_IOMODE_READ", 238
iparam, 172
language, 221
"MSK_LANG_ENG", 221
"MSK_LANG_DAN", 221
liinfitem, 233
"MSK_LIINF_RD_NUMQNZ", 234
"MSK_LIINF_RD_NUMANZ", 234
"MSK_LIINF_MIO_SIMPLEX_ITER", 234
"MSK_LIINF_MIO_SIM_MAXITER_SETBACKS", 234
"MSK_LIINF_MIO_PRESOLVED_ANZ", 234
"MSK_LIINF_MIO_INTPNT_ITER", 234
"MSK_LIINF_INTPNT_FACTOR_NUM_NZ", 233
"MSK_LIINF_BI_PRIMAL_ITER", 233
"MSK_LIINF_BI_DUAL_ITER", 233
"MSK_LIINF_BI_CLEAN_PRIMAL_ITER", 233
"MSK_LIINF_BI_CLEAN_PRIMAL_DEG_ITER", 233
"MSK_LIINF_BI_CLEAN_DUAL_ITER", 233
"MSK_LIINF_BI_CLEAN_DUAL_DEG_ITER", 233
mark, 221
"MSK_MARK_UP", 221
"MSK_MARK_LO", 221
miocontsoltype, 238
"MSK_MIO_CONT_SOL_ROOT", 238
"MSK_MIO_CONT_SOL_NONE", 238
"MSK_MIO_CONT_SOL_ITG_REL", 238
"MSK_MIO_CONT_SOL_ITG", 238
miomode, 238
"MSK_MIO_MODE_SATISFIED", 239
"MSK_MIO_MODE_IGNORED", 238
mionodeseltype, 239
"MSK_MIO_NODE_SELECTION_WORST", 239
"MSK_MIO_NODE_SELECTION_PSEUDO", 239
"MSK_MIO_NODE_SELECTION_HYBRID", 239
"MSK_MIO_NODE_SELECTION_FREE", 239
"MSK_MIO_NODE_SELECTION_FIRST", 239
"MSK_MIO_NODE_SELECTION_BEST", 239
mpsformat, 239
"MSK_MPS_FORMAT_STRICT", 239
"MSK_MPS_FORMAT_RELAXED", 239
"MSK_MPS_FORMAT_FREE", 239
"MSK_MPS_FORMAT_CPLEX", 239
nametype, 228
"MSK_NAME_TYPE_MPS", 228
"MSK_NAME_TYPE_LP", 228
"MSK_NAME_TYPE_GEN", 228
objsense, 239
"MSK_OBJECTIVE_SENSE_MINIMIZE", 239
"MSK_OBJECTIVE_SENSE_MAXIMIZE", 239
onoffkey, 239
"MSK_ON", 239
"MSK_OFF", 239
optimizertype, 239
"MSK_OPTIMIZER_PRIMAL_SIMPLEX", 240
"MSK_OPTIMIZER_MIXED_INT", 240
"MSK_OPTIMIZER_INTPNT", 240
"MSK_OPTIMIZER_FREE_SIMPLEX", 240
"MSK_OPTIMIZER_FREE", 239
"MSK_OPTIMIZER_DUAL_SIMPLEX", 239
"MSK_OPTIMIZER_CONIC", 239
orderingtype, 240
"MSK_ORDER_METHOD_TRY_GRAPHPAR", 240
"MSK_ORDER_METHOD_NONE", 240
"MSK_ORDER_METHOD_FREE", 240
"MSK_ORDER_METHOD_FORCE_GRAPHPAR", 240
"MSK_ORDER_METHOD_EXPERIMENTAL", 240
"MSK_ORDER_METHOD_APPMINLOC", 240
parametertype, 240
"MSK_PAR_STR_TYPE", 240
"MSK_PAR_INVALID_TYPE", 240
"MSK_PAR_INT_TYPE", 240
"MSK_PAR_DOU_TYPE", 240
presolvemode, 240
"MSK_PRESOLVE_MODE_ON", 240
"MSK_PRESOLVE_MODE_OFF", 240
"MSK_PRESOLVE_MODE_FREE", 240
problemitem, 240
"MSK_PI_VAR", 240
"MSK_PI_CONE", 241
"MSK_PI_CON", 240
problemtype, 241
"MSK_PROBTYPE_QO", 241
"MSK_PROBTYPE_QCQO", 241
"MSK_PROBTYPE_MIXED", 241
"MSK_PROBTYPE_LO", 241
"MSK_PROBTYPE_GECO", 241
"MSK_PROBTYPE_CONIC", 241
prosta, 241
"MSK_PRO_STA_UNKNOWN", 241
"MSK_PRO_STA_PRIM_INFEAS_OR_UNBOUNDED", 241
"MSK_PRO_STA_PRIM_INFEAS", 241
"MSK_PRO_STA_PRIM_FEAS", 241
"MSK_PRO_STA_PRIM_AND_DUAL_INFEAS", 241
"MSK_PRO_STA_PRIM_AND_DUAL_FEAS", 241
"MSK_PRO_STA_NEAR_PRIM_FEAS", 241
"MSK_PRO_STA_NEAR_PRIM_AND_DUAL_FEAS", 241
"MSK_PRO_STA_NEAR_DUAL_FEAS", 241
"MSK_PRO_STA_ILL_POSED", 241
"MSK_PRO_STA_DUAL_INFEAS", 241
"MSK_PRO_STA_DUAL_FEAS", 241
rescode, 199
rescodetype, 242

"MSK_RESPONSE_WRN", 242
"MSK_RESPONSE_UNK", 242
"MSK_RESPONSE_TRM", 242
"MSK_RESPONSE_OK", 242
"MSK_RESPONSE_ERR", 242
scalingmethod, 242
"MSK_SCALING_METHOD_POW2", 242
"MSK_SCALING_METHOD_FREE", 242
scalingtype, 242
"MSK_SCALING_NONE", 242
"MSK_SCALING_MODERATE", 242
"MSK_SCALING_FREE", 242
"MSK_SCALING_AGGRESSIVE", 242
sensitivitytype, 242
"MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION", 242
"MSK_SENSITIVITY_TYPE_BASIS", 242
simdegen, 221
"MSK_SIM_DEGEN_NONE", 222
"MSK_SIM_DEGEN_MODERATE", 222
"MSK_SIM_DEGEN_MINIMUM", 222
"MSK_SIM_DEGEN_FREE", 222
"MSK_SIM_DEGEN_AGGRESSIVE", 222
simdupvec, 222
"MSK_SIM_EXPLOIT_DUPVEC_ON", 222
"MSK_SIM_EXPLOIT_DUPVEC_OFF", 222
"MSK_SIM_EXPLOIT_DUPVEC_FREE", 222
simhotstart, 222
"MSK_SIM_HOTSTART_STATUS_KEYS", 223
"MSK_SIM_HOTSTART_NONE", 222
"MSK_SIM_HOTSTART_FREE", 222
simreform, 222
"MSK_SIM_REFORMULATION_ON", 222
"MSK_SIM_REFORMULATION_OFF", 222
"MSK_SIM_REFORMULATION_FREE", 222
"MSK_SIM_REFORMULATION_AGGRESSIVE", 222
simseltype, 242
"MSK_SIM_SELECTION_SE", 243
"MSK_SIM_SELECTION_PARTIAL", 243
"MSK_SIM_SELECTION_FULL", 242
"MSK_SIM_SELECTION_FREE", 242
"MSK_SIM_SELECTION_DEVEX", 243
"MSK_SIM_SELECTION_ASE", 242
solitem, 243
"MSK_SOL_ITEM_Y", 243
"MSK_SOL_ITEM_XX", 243
"MSK_SOL_ITEM_XC", 243
"MSK_SOL_ITEM_SUX", 243
"MSK_SOL_ITEM_SUC", 243
"MSK_SOL_ITEM_SNX", 243
"MSK_SOL_ITEM_SLX", 243
"MSK_SOL_ITEM_SLC", 243
solsta, 243
"MSK_SOL_STA_UNKNOWN", 243
"MSK_SOL_STA_PRIM_INFEAS_CER", 244
"MSK_SOL_STA_PRIM_ILLPOSED_CER", 244
"MSK_SOL_STA_PRIM_FEAS", 243
"MSK_SOL_STA_PRIM_AND_DUAL_FEAS", 243
"MSK_SOL_STA_OPTIMAL", 243
"MSK_SOL_STA_NEAR_PRIM_INFEAS_CER", 244
"MSK_SOL_STA_NEAR_PRIM_FEAS", 243
"MSK_SOL_STA_NEAR_PRIM_AND_DUAL_FEAS", 243
"MSK_SOL_STA_NEAR_OPTIMAL", 243
"MSK_SOL_STA_NEAR_INTEGER_OPTIMAL", 244
"MSK_SOL_STA_NEAR_DUAL_INFEAS_CER", 244
"MSK_SOL_STA_NEAR_DUAL_FEAS", 243
"MSK_SOL_STA_INTEGER_OPTIMAL", 244
"MSK_SOL_STA_DUAL_INFEAS_CER", 244
"MSK_SOL_STA_DUAL_ILLPOSED_CER", 244
"MSK_SOL_STA_DUAL_FEAS", 243
soltype, 244
"MSK_SOL_ITR", 244
"MSK_SOL_ITG", 244
"MSK_SOL_BAS", 244
solveform, 244
"MSK_SOLVE_PRIMAL", 244
"MSK_SOLVE_FREE", 244
"MSK_SOLVE_DUAL", 244
sparam, 196
stakey, 244
"MSK_SK_UPR", 244
"MSK_SK_UNK", 244
"MSK_SK_SUPBAS", 244
"MSK_SK_LOW", 244
"MSK_SK_INF", 245
"MSK_SK_FIX", 244
"MSK_SK_BAS", 244
startpointtype, 245
"MSK_STARTING_POINT_SATISFY_BOUNDS", 245
"MSK_STARTING_POINT_GUESS", 245
"MSK_STARTING_POINT_FREE", 245
"MSK_STARTING_POINT_CONSTANT", 245
streamtype, 245
"MSK_STREAM_WRN", 245
"MSK_STREAM_MSG", 245
"MSK_STREAM_LOG", 245
"MSK_STREAM_ERR", 245
symmattype, 228
"MSK_SYMMAT_TYPE_SPARSE", 228
transpose, 222
"MSK_TRANSPOSE_YES", 222
"MSK_TRANSPOSE_NO", 222
uplo, 222
"MSK_UPLO_UP", 222
"MSK_UPLO_LO", 222
value, 245
"MSK_MAX_STR_LEN", 245
"MSK_LICENSE_BUFFER_LENGTH", 245
variabletype, 245
"MSK_VAR_TYPE_INT", 245
"MSK_VAR_TYPE_CONT", 245
xmlwriteroutputtype, 241
"MSK_WRITE_XML_MODE_ROW", 242
"MSK_WRITE_XML_MODE_COL", 242

## Functions

intlinprog, 137
linprog, 137
lsqlin, 139
lsqnonneg, 140
mosekopt, 132
mskenopt, 134
mskgpopt, 134
mskgpread, 135
mskgpwri, 135
msklpopt, 133
mskoptimget, 136
mskoptimset, 136
mskqpopt, 133
mskscopt, 134
quadprog, 141

## Parameters

Double parameters, 162
MSK_DPAR_ANA_SOL_INFEAS_TOL, 162
MSK_DPAR_BASIS_REL_TOL_S, 162
MSK_DPAR_BASIS_TOL_S, 162
MSK_DPAR_BASIS_TOL_X, 162
MSK_DPAR_CHECK_CONVEXITY_REL_TOL, 163
MSK_DPAR_DATA_SYM_MAT_TOL, 163
MSK_DPAR_DATA_SYM_MAT_TOL_HUGE, 163
MSK_DPAR_DATA_SYM_MAT_TOL_LARGE, 163
MSK_DPAR_DATA_TOL_AIJ, 163
MSK_DPAR_DATA_TOL_AIJ_HUGE, 163
MSK_DPAR_DATA_TOL_AIJ_LARGE, 163
MSK_DPAR_DATA_TOL_BOUND_INF, 164
MSK_DPAR_DATA_TOL_BOUND_WRN, 164
MSK_DPAR_DATA_TOL_C_HUGE, 164
MSK_DPAR_DATA_TOL_CJ_LARGE, 164
MSK_DPAR_DATA_TOL_QIJ, 164
MSK_DPAR_DATA_TOL_X, 164
MSK_DPAR_INTPNT_CO_TOL_DFEAS, 164
MSK_DPAR_INTPNT_CO_TOL_INFEAS, 165
MSK_DPAR_INTPNT_CO_TOL_MU_RED, 165
MSK_DPAR_INTPNT_CO_TOL_NEAR_REL, 165
MSK_DPAR_INTPNT_CO_TOL_PFEAS, 165
MSK_DPAR_INTPNT_CO_TOL_REL_GAP, 165
MSK_DPAR_INTPNT_NL_MERIT_BAL, 165
MSK_DPAR_INTPNT_NL_TOL_DFEAS, 165
MSK_DPAR_INTPNT_NL_TOL_MU_RED, 165
MSK_DPAR_INTPNT_NL_TOL_NEAR_REL, 166
MSK_DPAR_INTPNT_NL_TOL_PFEAS, 166
MSK_DPAR_INTPNT_NL_TOL_REL_GAP, 166
MSK_DPAR_INTPNT_NL_TOL_REL_STEP, 166
MSK_DPAR_INTPNT_QO_TOL_DFEAS, 166
MSK_DPAR_INTPNT_QO_TOL_INFEAS, 166
MSK_DPAR_INTPNT_QO_TOL_MU_RED, 166
MSK_DPAR_INTPNT_QO_TOL_NEAR_REL, 167
MSK_DPAR_INTPNT_QO_TOL_PFEAS, 167
MSK_DPAR_INTPNT_QO_TOL_REL_GAP, 167
MSK_DPAR_INTPNT_TOL_DFEAS, 167
MSK_DPAR_INTPNT_TOL_DSAFE, 167
MSK_DPAR_INTPNT_TOL_INFEAS, 167
MSK_DPAR_INTPNT_TOL_MU_RED, 167
MSK_DPAR_INTPNT_TOL_PATH, 168
MSK_DPAR_INTPNT_TOL_PFEAS, 168
MSK_DPAR_INTPNT_TOL_PSAFE, 168
MSK_DPAR_INTPNT_TOL_REL_GAP, 168
MSK_DPAR_INTPNT_TOL_REL_STEP, 168
MSK_DPAR_INTPNT_TOL_STEP_SIZE, 168
MSK_DPAR_LOWER_OBJ_CUT, 168
MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH, 169
MSK_DPAR_MIO_DISABLE_TERM_TIME, 169
MSK_DPAR_MIO_MAX_TIME, 169
MSK_DPAR_MIO_NEAR_TOL_ABS_GAP, 169
MSK_DPAR_MIO_NEAR_TOL_REL_GAP, 169
MSK_DPAR_MIO_REL_GAP_CONST, 170
MSK_DPAR_MIO_TOL_ABS_GAP, 170
MSK_DPAR_MIO_TOL_ABS_RELAX_INT, 170
MSK_DPAR_MIO_TOL_FEAS, 170
MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT, 170
MSK_DPAR_MIO_TOL_REL_GAP, 170
MSK_DPAR_OPTIMIZER_MAX_TIME, 170
MSK_DPAR_PRESOLVE_TOL_ABS_LINDEP, 170
MSK_DPAR_PRESOLVE_TOL_AIJ, 171
MSK_DPAR_PRESOLVE_TOL_REL_LINDEP, 171
MSK_DPAR_PRESOLVE_TOL_S, 171
MSK_DPAR_PRESOLVE_TOL_X, 171
MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL, 171
MSK_DPAR_SEMIDEFINITE_TOL_APPROX, 171
MSK_DPAR_SIM_LU_TOL_REL_PIV, 171
MSK_DPAR_SIMPLEX_ABS_TOL_PIV, 171
MSK_DPAR_UPPER_OBJ_CUT, 172
MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH, 172
Integer parameters, 172
MSK_IPAR_ANA_SOL_BASIS, 172
MSK_IPAR_ANA_SOL_PRINT_VIOLATED, 172
MSK_IPAR_AUTO_SORT_A_BEFORE_OPT, 172
MSK_IPAR_AUTO_UPDATE_SOL_INFO, 172
MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE, 173
MSK_IPAR_BI_CLEAN_OPTIMIZER, 173
MSK_IPAR_BI_IGNORE_MAX_ITER, 173
MSK_IPAR_BI_IGNORE_NUM_ERROR, 173
MSK_IPAR_BI_MAX_ITERATIONS, 173
MSK_IPAR_CACHE_LICENSE, 173
MSK_IPAR_CHECK_CONVEXITY, 173
MSK_IPAR_COMPRESS_STATFILE, 174
MSK_IPAR_INFEAS_GENERIC_NAMES, 174
MSK_IPAR_INFEAS_PREFER_PRIMAL, 174
MSK_IPAR_INFEAS_REPORT_AUTO, 174
MSK_IPAR_INFEAS_REPORT_LEVEL, 174
MSK_IPAR_INTPNT_BASIS, 174
MSK_IPAR_INTPNT_DIFF_STEP, 174
MSK_IPAR_INTPNT_HOTSTART, 175
MSK_IPAR_INTPNT_MAX_ITERATIONS, 175
MSK_IPAR_INTPNT_MAX_NUM_COR, 175
MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS, 175
MSK_IPAR_INTPNT_MULTI_THREAD, 175
MSK_IPAR_INTPNT_OFF_COL_TRH, 175

MSK_IPAR_INTPNT_ORDER_METHOD, 175
MSK_IPAR_INTPNT_REGULARIZATION_USE, 176
MSK_IPAR_INTPNT_SCALING, 176
MSK_IPAR_INTPNT_SOLVE_FORM, 176
MSK_IPAR_INTPNT_STARTING_POINT, 176
MSK_IPAR_LICENSE_DEBUG, 176
MSK_IPAR_LICENSE_PAUSE_TIME, 176
MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS, 176
MSK_IPAR_LICENSE_TRH_EXPIRY_WRN, 176
MSK_IPAR_LICENSE_WAIT, 177
MSK_IPAR_LOG, 177
MSK_IPAR_LOG_ANA_PRO, 177
MSK_IPAR_LOG_BI, 177
MSK_IPAR_LOG_BI_FREQ, 177
MSK_IPAR_LOG_CHECK_CONVEXITY, 177
MSK_IPAR_LOG_CUT_SECOND_OPT, 177
MSK_IPAR_LOG_EXPAND, 178
MSK_IPAR_LOG_FEAS_REPAIR, 178
MSK_IPAR_LOG_FILE, 178
MSK_IPAR_LOG_INFEAS_ANA, 178
MSK_IPAR_LOG_INTPNT, 178
MSK_IPAR_LOG_MIO, 178
MSK_IPAR_LOG_MIO_FREQ, 178
MSK_IPAR_LOG_ORDER, 179
MSK_IPAR_LOG_PRESOLVE, 179
MSK_IPAR_LOG_RESPONSE, 179
MSK_IPAR_LOG_SENSITIVITY, 179
MSK_IPAR_LOG_SENSITIVITY_OPT, 179
MSK_IPAR_LOG_SIM, 179
MSK_IPAR_LOG_SIM_FREQ, 180
MSK_IPAR_LOG_SIM_MINOR, 180
MSK_IPAR_LOG_STORAGE, 180
MSK_IPAR_MAX_NUM_WARNINGS, 180
MSK_IPAR_MIO_BRANCH_DIR, 180
MSK_IPAR_MIO_CONSTRUCT_SOL, 180
MSK_IPAR_MIO_CUT_CLIQUE, 180
MSK_IPAR_MIO_CUT_CMIR, 180
MSK_IPAR_MIO_CUT_GMI, 181
MSK_IPAR_MIO_CUT_IMPLIED_BOUND, 181
MSK_IPAR_MIO_CUT_KNAPSACK_COVER, 181
MSK_IPAR_MIO_CUT_SELECTION_LEVEL, 181
MSK_IPAR_MIO_HEURISTIC_LEVEL, 181
MSK_IPAR_MIO_MAX_NUM_BRANCHES, 182
MSK_IPAR_MIO_MAX_NUM_RELAXS, 182
MSK_IPAR_MIO_MAX_NUM_SOLUTIONS, 182
MSK_IPAR_MIO_MODE, 182
MSK_IPAR_MIO_MT_USER_CB, 182
MSK_IPAR_MIO_NODE_OPTIMIZER, 182
MSK_IPAR_MIO_NODE_SELECTION, 182
MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE, 183
MSK_IPAR_MIO_PROBING_LEVEL, 183
MSK_IPAR_MIO_RINS_MAX_NODES, 183
MSK_IPAR_MIO_ROOT_OPTIMIZER, 183
MSK_IPAR_MIO_ROOT_REPEAT_PRESOLVE_LEVEL, 183
MSK_IPAR_MIO_VB_DETECTION_LEVEL, 183
MSK_IPAR_MT_SPINCOUNT, 184
MSK_IPAR_NUM_THREADS, 184
MSK_IPAR_OPF_MAX_TERMS_PER_LINE, 184
MSK_IPAR_OPF_WRITE_HEADER, 184
MSK_IPAR_OPF_WRITE_HINTS, 184
MSK_IPAR_OPF_WRITE_PARAMETERS, 184
MSK_IPAR_OPF_WRITE_PROBLEM, 184
MSK_IPAR_OPF_WRITE_SOL_BAS, 185
MSK_IPAR_OPF_WRITE_SOL_ITG, 185
MSK_IPAR_OPF_WRITE_SOL_ITR, 185
MSK_IPAR_OPF_WRITE_SOLUTIONS, 185
MSK_IPAR_OPTIMIZER, 185
MSK_IPAR_PARAM_READ_CASE_NAME, 185
MSK_IPAR_PARAM_READ_IGN_ERROR, 185
MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_FILL, 186
MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES, 186
MSK_IPAR_PRESOLVE_LEVEL, 186
MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH, 186
MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH, 186
MSK_IPAR_PRESOLVE_LINDEP_USE, 186
MSK_IPAR_PRESOLVE_MAX_NUM_REDUCTIONS, 186
MSK_IPAR_PRESOLVE_USE, 186
MSK_IPAR_PRIMAL_REPAIR_OPTIMIZER, 187
MSK_IPAR_READ_DATA_COMPRESSED, 187
MSK_IPAR_READ_DATA_FORMAT, 187
MSK_IPAR_READ_DEBUG, 187
MSK_IPAR_READ_KEEP_FREE_CON, 187
MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU, 187
MSK_IPAR_READ_LP_QUOTED_NAMES, 187
MSK_IPAR_READ_MPS_FORMAT, 188
MSK_IPAR_READ_MPS_WIDTH, 188
MSK_IPAR_READ_TASK_IGNORE_PARAM, 188
MSK_IPAR_REMOVE_UNUSED_SOLUTIONS, 188
MSK_IPAR_SENSITIVITY_ALL, 188
MSK_IPAR_SENSITIVITY_OPTIMIZER, 188
MSK_IPAR_SENSITIVITY_TYPE, 188
MSK_IPAR_SIM_BASIS_FACTOR_USE, 188
MSK_IPAR_SIM_DEGEN, 189
MSK_IPAR_SIM_DUAL_CRASH, 189
MSK_IPAR_SIM_DUAL_PHASEONE_METHOD, 189
MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION, 189
MSK_IPAR_SIM_DUAL_SELECTION, 189
MSK_IPAR_SIM_EXPLOIT_DUPVEC, 189
MSK_IPAR_SIM_HOTSTART, 189
MSK_IPAR_SIM_HOTSTART_LU, 190
MSK_IPAR_SIM_MAX_ITERATIONS, 190
MSK_IPAR_SIM_MAX_NUM_SETBACKS, 190
MSK_IPAR_SIM_NON_SINGULAR, 190
MSK_IPAR_SIM_PRIMAL_CRASH, 190
MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD, 190
MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION, 190
MSK_IPAR_SIM_PRIMAL_SELECTION, 191
MSK_IPAR_SIM_REFACTOR_FREQ, 191
MSK_IPAR_SIM_REFORMULATION, 191
MSK_IPAR_SIM_SAVE_LU, 191
MSK_IPAR_SIM_SCALING, 191
MSK_IPAR_SIM_SCALING_METHOD, 191
MSK_IPAR_SIM_SOLVE_FORM, 191
MSK_IPAR_SIM_STABILITY_PRIORITY, 192

MSK_IPAR_SIM_SWITCH_OPTIMIZER, 192
MSK_IPAR_SOL_FILTER_KEEP_BASIC, 192
MSK_IPAR_SOL_FILTER_KEEP_RANGED, 192
MSK_IPAR_SOL_READ_NAME_WIDTH, 192
MSK_IPAR_SOL_READ_WIDTH, 192
MSK_IPAR_SOLUTION_CALLBACK, 192
MSK_IPAR_TIMING_LEVEL, 193
MSK_IPAR_WRITE_BAS_CONSTRAINTS, 193
MSK_IPAR_WRITE_BAS_HEAD, 193
MSK_IPAR_WRITE_BAS_VARIABLES, 193
MSK_IPAR_WRITE_DATA_COMPRESSED, 193
MSK_IPAR_WRITE_DATA_FORMAT, 193
MSK_IPAR_WRITE_DATA_PARAM, 193
MSK_IPAR_WRITE_FREE_CON, 193
MSK_IPAR_WRITE_GENERIC_NAMES, 194
MSK_IPAR_WRITE_GENERIC_NAMES_IO, 194
MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_ITEMS, 194
MSK_IPAR_WRITE_INT_CONSTRAINTS, 194
MSK_IPAR_WRITE_INT_HEAD, 194
MSK_IPAR_WRITE_INT_VARIABLES, 194
MSK_IPAR_WRITE_LP_FULL_OBJ, 194
MSK_IPAR_WRITE_LP_LINE_WIDTH, 195
MSK_IPAR_WRITE_LP_QUOTED_NAMES, 195
MSK_IPAR_WRITE_LP_STRICT_FORMAT, 195
MSK_IPAR_WRITE_LP_TERMS_PER_LINE, 195
MSK_IPAR_WRITE_MPS_FORMAT, 195
MSK_IPAR_WRITE_MPS_INT, 195
MSK_IPAR_WRITE_PRECISION, 195
MSK_IPAR_WRITE_SOL_BARVARIABLES, 195
MSK_IPAR_WRITE_SOL_CONSTRAINTS, 196
MSK_IPAR_WRITE_SOL_HEAD, 196
MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES, 196
MSK_IPAR_WRITE_SOL_VARIABLES, 196
MSK_IPAR_WRITE_TASK_INC_SOL, 196
MSK_IPAR_WRITE_XML_MODE, 196
String parameters, 196
MSK_SPAR_BAS_SOL_FILE_NAME, 196
MSK_SPAR_DATA_FILE_NAME, 197
MSK_SPAR_DEBUG_FILE_NAME, 197
MSK_SPAR_INT_SOL_FILE_NAME, 197
MSK_SPAR_ITR_SOL_FILE_NAME, 197
MSK_SPAR_MIO_DEBUG_STRING, 197
MSK_SPAR_PARAM_COMMENT_SIGN, 197
MSK_SPAR_PARAM_READ_FILE_NAME, 197
MSK_SPAR_PARAM_WRITE_FILE_NAME, 197
MSK_SPAR_READ_MPS_BOU_NAME, 197
MSK_SPAR_READ_MPS_OBJ_NAME, 197
MSK_SPAR_READ_MPS_RAN_NAME, 198
MSK_SPAR_READ_MPS_RHS_NAME, 198
MSK_SPAR_REMOTE_ACCESS_TOKEN, 198
MSK_SPAR_SENSITIVITY_FILE_NAME, 198
MSK_SPAR_SENSITIVITY_RES_FILE_NAME, 198
MSK_SPAR_SOL_FILTER_XC_LOW, 198
MSK_SPAR_SOL_FILTER_XC_UPR, 198
MSK_SPAR_SOL_FILTER_XX_LOW, 198
MSK_SPAR_SOL_FILTER_XX_UPR, 198
MSK_SPAR_STAT_FILE_NAME, 199
MSK_SPAR_STAT_KEY, 199
MSK_SPAR_STAT_NAME, 199
MSK_SPAR_WRITE_LP_GEN_VAR_NAME, 199

## Response codes

Termination, 199
"MSK_RES_OK", 199
"MSK_RES_TRM_INTERNAL", 200
"MSK_RES_TRM_INTERNAL_STOP", 200
"MSK_RES_TRM_MAX_ITERATIONS", 199
"MSK_RES_TRM_MAX_NUM_SETBACKS", 200
"MSK_RES_TRM_MAX_TIME", 199
"MSK_RES_TRM_MIO_NEAR_ABS_GAP", 200
"MSK_RES_TRM_MIO_NEAR_REL_GAP", 199
"MSK_RES_TRM_MIO_NUM_BRANCHES", 200
"MSK_RES_TRM_MIO_NUM_RELAXS", 200
"MSK_RES_TRM_NUM_MAX_NUM_INT_SOLUTIONS", 200
"MSK_RES_TRM_NUMERICAL_PROBLEM", 200
"MSK_RES_TRM_OBJECTIVE_RANGE", 199
"MSK_RES_TRM_STALL", 200
"MSK_RES_TRM_USER_CALLBACK", 200
Warnings, 200
"MSK_RES_WRN_ANA_ALMOST_INT_BOUNDS", 203
"MSK_RES_WRN_ANA_C_ZERO", 203
"MSK_RES_WRN_ANA_CLOSE_BOUNDS", 203
"MSK_RES_WRN_ANA_EMPTY_COLS", 203
"MSK_RES_WRN_ANA_LARGE_BOUNDS", 203
"MSK_RES_WRN_CONSTRUCT_INVALID_SOL_ITG", 202
"MSK_RES_WRN_CONSTRUCT_NO_SOL_ITG", 203
"MSK_RES_WRN_CONSTRUCT_SOLUTION_INFEAS", 202
"MSK_RES_WRN_DROPPED_NZ_QOBJ", 201
"MSK_RES_WRN_DUPLICATE_BARVARIABLE_NAMES", 203
"MSK_RES_WRN_DUPLICATE_CONE_NAMES", 203
"MSK_RES_WRN_DUPLICATE_CONSTRAINT_NAMES", 203
"MSK_RES_WRN_DUPLICATE_VARIABLE_NAMES", 203
"MSK_RES_WRN_ELIMINATOR_SPACE", 202
"MSK_RES_WRN_EMPTY_NAME", 202
"MSK_RES_WRN_IGNORE_INTEGER", 201
"MSK_RES_WRN_INCOMPLETE_LINEAR_DEPENDENCY_CHECK", 202
"MSK_RES_WRN_LARGE_AIJ", 201
"MSK_RES_WRN_LARGE_BOUND", 200
"MSK_RES_WRN_LARGE_CJ", 201
"MSK_RES_WRN_LARGE_CON_FX", 201
"MSK_RES_WRN_LARGE_LO_BOUND", 200
"MSK_RES_WRN_LARGE_UP_BOUND", 200
"MSK_RES_WRN_LICENSE_EXPIRE", 202
"MSK_RES_WRN_LICENSE_FEATURE_EXPIRE", 202
"MSK_RES_WRN_LICENSE_SERVER", 202
"MSK_RES_WRN_LP_DROP_VARIABLE", 201
"MSK_RES_WRN_LP_OLD_QUAD_FORMAT", 201
"MSK_RES_WRN_MIO_INFEASIBLE_FINAL", 201

"MSK_RES_WRN_MPS_SPLIT_BOU_VECTOR", 201
"MSK_RES_WRN_MPS_SPLIT_RAN_VECTOR", 201
"MSK_RES_WRN_MPS_SPLIT_RHS_VECTOR", 201
"MSK_RES_WRN_NAME_MAX_LEN", 201
"MSK_RES_WRN_NO_DUALIZER", 203
"MSK_RES_WRN_NO_GLOBAL_OPTIMIZER", 201
"MSK_RES_WRN_NO_NONLINEAR_FUNCTION_WRITE", 202
"MSK_RES_WRN_NZ_IN_UPR_TRI", 201
"MSK_RES_WRN_OPEN_PARAM_FILE", 200
"MSK_RES_WRN_PARAM_IGNORED_CMIO", 202
"MSK_RES_WRN_PARAM_NAME_DOU", 202
"MSK_RES_WRN_PARAM_NAME_INT", 202
"MSK_RES_WRN_PARAM_NAME_STR", 202
"MSK_RES_WRN_PARAM_STR_VALUE", 202
"MSK_RES_WRN_PRESOLVE_OUTOFSPACE", 202
"MSK_RES_WRN_QUAD_CONES_WITH_ROOT_FIXED_AT_ZERO", 203
"MSK_RES_WRN_RQUAD_CONES_WITH_ROOT_FIXED_AT_ZERO", 203
"MSK_RES_WRN_SOL_FILE_IGNORED_CON", 201
"MSK_RES_WRN_SOL_FILE_IGNORED_VAR", 201
"MSK_RES_WRN_SOL_FILTER", 201
"MSK_RES_WRN_SPAR_MAX_LEN", 201
"MSK_RES_WRN_SYM_MAT_LARGE", 203
"MSK_RES_WRN_TOO_FEW_BASIS_VARS", 201
"MSK_RES_WRN_TOO_MANY_BASIS_VARS", 201
"MSK_RES_WRN_UNDEF_SOL_FILE_NAME", 201
"MSK_RES_WRN_USING_GENERIC_NAMES", 202
"MSK_RES_WRN_WRITE_CHANGED_NAMES", 202
"MSK_RES_WRN_WRITE_DISCARDED_CFIX", 202
"MSK_RES_WRN_ZERO_AIJ", 201
"MSK_RES_WRN_ZEROS_IN_SPARSE_COL", 202
"MSK_RES_WRN_ZEROS_IN_SPARSE_ROW", 202
Errors, 203
"MSK_RES_ERR_AD_INVALID_CODELIST", 217
"MSK_RES_ERR_API_ARRAY_TOO_SMALL", 216
"MSK_RES_ERR_API_CB_CONNECT", 216
"MSK_RES_ERR_API_FATAL_ERROR", 216
"MSK_RES_ERR_API_INTERNAL", 216
"MSK_RES_ERR_ARG_IS_TOO_LARGE", 209
"MSK_RES_ERR_ARG_IS_TOO_SMALL", 209
"MSK_RES_ERR_ARGUMENT_DIMENSION", 209
"MSK_RES_ERR_ARGUMENT_IS_TOO_LARGE", 218
"MSK_RES_ERR_ARGUMENT_LENNEQ", 208
"MSK_RES_ERR_ARGUMENT_PERM_ARRAY", 212
"MSK_RES_ERR_ARGUMENT_TYPE", 209
"MSK_RES_ERR_BAR_VAR_DIM", 217
"MSK_RES_ERR_BASIS", 211
"MSK_RES_ERR_BASIS_FACTOR", 215
"MSK_RES_ERR_BASIS_SINGULAR", 215
"MSK_RES_ERR_BLANK_NAME", 206
"MSK_RES_ERR_CANNOT_CLONE_NL", 216
"MSK_RES_ERR_CANNOT_HANDLE_NL", 216
"MSK_RES_ERR_CBF_DUPLICATE_ACOORD", 220
"MSK_RES_ERR_CBF_DUPLICATE_BCOORD", 220
"MSK_RES_ERR_CBF_DUPLICATE_CON", 219
"MSK_RES_ERR_CBF_DUPLICATE_INT", 219
"MSK_RES_ERR_CBF_DUPLICATE_OBJ", 219
"MSK_RES_ERR_CBF_DUPLICATE_OBJACOORD", 219
"MSK_RES_ERR_CBF_DUPLICATE_PSDVAR", 220
"MSK_RES_ERR_CBF_DUPLICATE_VAR", 219
"MSK_RES_ERR_CBF_INVALID_CON_TYPE", 219
"MSK_RES_ERR_CBF_INVALID_DOMAIN_DIMENSION", 219
"MSK_RES_ERR_CBF_INVALID_INT_INDEX", 220
"MSK_RES_ERR_CBF_INVALID_PSDVAR_DIMENSION", 220
"MSK_RES_ERR_CBF_INVALID_VAR_TYPE", 219
"MSK_RES_ERR_CBF_NO_VARIABLES", 219
"MSK_RES_ERR_CBF_NO_VERSION_SPECIFIED", 219
"MSK_RES_ERR_CBF_OBJ_SENSE", 219
"MSK_RES_ERR_CBF_PARSE", 219
"MSK_RES_ERR_CBF_SYNTAX", 219
"MSK_RES_ERR_CBF_TOO_FEW_CONSTRAINTS", 220
"MSK_RES_ERR_CBF_TOO_FEW_INTS", 220
"MSK_RES_ERR_CBF_TOO_FEW_PSDVAR", 220
"MSK_RES_ERR_CBF_TOO_FEW_VARIABLES", 220
"MSK_RES_ERR_CBF_TOO_MANY_CONSTRAINTS", 219
"MSK_RES_ERR_CBF_TOO_MANY_INTS", 220
"MSK_RES_ERR_CBF_TOO_MANY_VARIABLES", 219
"MSK_RES_ERR_CBF_UNSUPPORTED", 220
"MSK_RES_ERR_CON_Q_NOT_NSD", 212
"MSK_RES_ERR_CON_Q_NOT_PSD", 212
"MSK_RES_ERR_CONE_INDEX", 212
"MSK_RES_ERR_CONE_OVERLAP", 212
"MSK_RES_ERR_CONE_OVERLAP_APPEND", 213
"MSK_RES_ERR_CONE_REP_VAR", 213
"MSK_RES_ERR_CONE_SIZE", 212
"MSK_RES_ERR_CONE_TYPE", 213
"MSK_RES_ERR_CONE_TYPE_STR", 213
"MSK_RES_ERR_DATA_FILE_EXT", 205
"MSK_RES_ERR_DUP_NAME", 206
"MSK_RES_ERR_DUPLICATE_AIJ", 213
"MSK_RES_ERR_DUPLICATE_BARVARIABLE_NAMES", 218
"MSK_RES_ERR_DUPLICATE_CONE_NAMES", 218
"MSK_RES_ERR_DUPLICATE_CONSTRAINT_NAMES", 218
"MSK_RES_ERR_DUPLICATE_VARIABLE_NAMES", 218
"MSK_RES_ERR_END_OF_FILE", 205
"MSK_RES_ERR_FACTOR", 215
"MSK_RES_ERR_FEASREPAIR_CANNOT_RELAX", 215
"MSK_RES_ERR_FEASREPAIR_INCONSISTENT_BOUND", 215
"MSK_RES_ERR_FEASREPAIR_SOLVING_RELAXED", 215
"MSK_RES_ERR_FILE_LICENSE", 204
"MSK_RES_ERR_FILE_OPEN", 205
"MSK_RES_ERR_FILE_READ", 205
"MSK_RES_ERR_FILE_WRITE", 205
"MSK_RES_ERR_FINAL_SOLUTION", 215
"MSK_RES_ERR_FIRST", 211
"MSK_RES_ERR_FIRSTI", 212
"MSK_RES_ERR_FIRSTJ", 212
"MSK_RES_ERR_FIXED_BOUND_VALUES", 214

"MSK_RES_ERR_FLEXLM", 204
"MSK_RES_ERR_GLOBAL_INV_CONIC_PROBLEM", 215
"MSK_RES_ERR_HUGE_AIJ", 213
"MSK_RES_ERR_HUGE_C", 213
"MSK_RES_ERR_IDENTICAL_TASKS", 217
"MSK_RES_ERR_IN_ARGUMENT", 209
"MSK_RES_ERR_INDEX", 210
"MSK_RES_ERR_INDEX_ARR_IS_TOO_LARGE", 209
"MSK_RES_ERR_INDEX_ARR_IS_TOO_SMALL", 209
"MSK_RES_ERR_INDEX_IS_TOO_LARGE", 209
"MSK_RES_ERR_INDEX_IS_TOO_SMALL", 209
"MSK_RES_ERR_INF_DOU_INDEX", 209
"MSK_RES_ERR_INF_DOU_NAME", 210
"MSK_RES_ERR_INF_INT_INDEX", 209
"MSK_RES_ERR_INF_INT_NAME", 210
"MSK_RES_ERR_INF_LINT_INDEX", 209
"MSK_RES_ERR_INF_LINT_NAME", 210
"MSK_RES_ERR_INF_TYPE", 210
"MSK_RES_ERR_INFEAS_UNDEFINED", 217
"MSK_RES_ERR_INFINITE_BOUND", 213
"MSK_RES_ERR_INT64_TO_INT32_CAST", 217
"MSK_RES_ERR_INTERNAL", 216
"MSK_RES_ERR_INTERNAL_TEST_FAILED", 217
"MSK_RES_ERR_INV_APTRE", 210
"MSK_RES_ERR_INV_BK", 210
"MSK_RES_ERR_INV_BKC", 211
"MSK_RES_ERR_INV_BKX", 211
"MSK_RES_ERR_INV_CONE_TYPE", 211
"MSK_RES_ERR_INV_CONE_TYPE_STR", 211
"MSK_RES_ERR_INV_MARKI", 216
"MSK_RES_ERR_INV_MARKJ", 216
"MSK_RES_ERR_INV_NAME_ITEM", 212
"MSK_RES_ERR_INV_NUMI", 216
"MSK_RES_ERR_INV_NUMJ", 216
"MSK_RES_ERR_INV_OPTIMIZER", 215
"MSK_RES_ERR_INV_PROBLEM", 215
"MSK_RES_ERR_INV_QCON_SUBI", 213
"MSK_RES_ERR_INV_QCON_SUBJ", 213
"MSK_RES_ERR_INV_QCON_SUBK", 213
"MSK_RES_ERR_INV_QCON_VAL", 213
"MSK_RES_ERR_INV_QOBJ_SUBI", 213
"MSK_RES_ERR_INV_QOBJ_SUBJ", 213
"MSK_RES_ERR_INV_QOBJ_VAL", 213
"MSK_RES_ERR_INV_SK", 211
"MSK_RES_ERR_INV_SK_STR", 211
"MSK_RES_ERR_INV_SKC", 211
"MSK_RES_ERR_INV_SKN", 211
"MSK_RES_ERR_INV_SKX", 211
"MSK_RES_ERR_INV_VAR_TYPE", 211
"MSK_RES_ERR_INVALID_ACCMODE", 216
"MSK_RES_ERR_INVALID_AIJ", 214
"MSK_RES_ERR_INVALID_AMPL_STUB", 217
"MSK_RES_ERR_INVALID_BARVAR_NAME", 206
"MSK_RES_ERR_INVALID_COMPRESSION", 215
"MSK_RES_ERR_INVALID_CON_NAME", 206
"MSK_RES_ERR_INVALID_CONE_NAME", 206
"MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_CONES", 218
"MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_GENERAL_NL", 218
"MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_SYM_MAT", 218
"MSK_RES_ERR_INVALID_FILE_NAME", 205
"MSK_RES_ERR_INVALID_FORMAT_TYPE", 212
"MSK_RES_ERR_INVALID_IDX", 210
"MSK_RES_ERR_INVALID_IOMODE", 215
"MSK_RES_ERR_INVALID_MAX_NUM", 210
"MSK_RES_ERR_INVALID_NAME_IN_SOL_FILE", 208
"MSK_RES_ERR_INVALID_OBJ_NAME", 206
"MSK_RES_ERR_INVALID_OBJECTIVE_SENSE", 214
"MSK_RES_ERR_INVALID_PROBLEM_TYPE", 218
"MSK_RES_ERR_INVALID_SOL_FILE_NAME", 205
"MSK_RES_ERR_INVALID_STREAM", 205
"MSK_RES_ERR_INVALID_SURPLUS", 211
"MSK_RES_ERR_INVALID_SYM_MAT_DIM", 218
"MSK_RES_ERR_INVALID_TASK", 206
"MSK_RES_ERR_INVALID_UTF8", 216
"MSK_RES_ERR_INVALID_VAR_NAME", 206
"MSK_RES_ERR_INVALID_WCHAR", 216
"MSK_RES_ERR_INVALID_WHICHSOL", 209
"MSK_RES_ERR_JSON_DATA", 208
"MSK_RES_ERR_JSON_FORMAT", 208
"MSK_RES_ERR_JSON_MISSING_DATA", 208
"MSK_RES_ERR_JSON_NUMBER_OVERFLOW", 208
"MSK_RES_ERR_JSON_STRING", 208
"MSK_RES_ERR_JSON_SYNTAX", 208
"MSK_RES_ERR_LAST", 211
"MSK_RES_ERR_LASTI", 212
"MSK_RES_ERR_LASTJ", 212
"MSK_RES_ERR_LAU_ARG_K", 219
"MSK_RES_ERR_LAU_ARG_M", 219
"MSK_RES_ERR_LAU_ARG_N", 219
"MSK_RES_ERR_LAU_ARG_TRANS", 219
"MSK_RES_ERR_LAU_ARG_TRANSA", 219
"MSK_RES_ERR_LAU_ARG_TRANSB", 219
"MSK_RES_ERR_LAU_ARG_UPLO", 219
"MSK_RES_ERR_LAU_INVALID_LOWER_TRIANGULAR_MATRIX", 218
"MSK_RES_ERR_LAU_INVALID_SPARSE_SYMMETRIC_MATRIX", 219
"MSK_RES_ERR_LAU_NOT_POSITIVE_DEFINITE", 218
"MSK_RES_ERR_LAU_SINGULAR_MATRIX", 218
"MSK_RES_ERR_LAU_UNKNOWN", 218
"MSK_RES_ERR_LICENSE", 203
"MSK_RES_ERR_LICENSE_CANNOT_ALLOCATE", 204
"MSK_RES_ERR_LICENSE_CANNOT_CONNECT", 204
"MSK_RES_ERR_LICENSE_EXPIRED", 203
"MSK_RES_ERR_LICENSE_FEATURE", 204
"MSK_RES_ERR_LICENSE_INVALID_HOSTID", 204
"MSK_RES_ERR_LICENSE_MAX", 204
"MSK_RES_ERR_LICENSE_MOSEKLM_DAEMON", 204
"MSK_RES_ERR_LICENSE_NO_SERVER_LINE", 204
"MSK_RES_ERR_LICENSE_NO_SERVER_SUPPORT", 204
"MSK_RES_ERR_LICENSE_SERVER", 204

"MSK_RES_ERR_LICENSE_SERVER_VERSION", 204
"MSK_RES_ERR_LICENSE_VERSION", 203
"MSK_RES_ERR_LINK_FILE_DLL", 205
"MSK_RES_ERR_LIVING_TASKS", 206
"MSK_RES_ERR_LOWER_BOUND_IS_A_NAN", 213
"MSK_RES_ERR_LP_DUP_SLACK_NAME", 207
"MSK_RES_ERR_LP_EMPTY", 207
"MSK_RES_ERR_LP_FILE_FORMAT", 208
"MSK_RES_ERR_LP_FORMAT", 208
"MSK_RES_ERR_LP_FREE_CONSTRAINT", 208
"MSK_RES_ERR_LP_INCOMPATIBLE", 207
"MSK_RES_ERR_LP_INVALID_CON_NAME", 208
"MSK_RES_ERR_LP_INVALID_VAR_NAME", 207
"MSK_RES_ERR_LP_WRITE_CONIC_PROBLEM", 208
"MSK_RES_ERR_LP_WRITE_GECO_PROBLEM", 208
"MSK_RES_ERR_LU_MAX_NUM_TRIES", 216
"MSK_RES_ERR_MAX_LEN_IS_TOO_SMALL", 212
"MSK_RES_ERR_MAXNUMBARVAR", 210
"MSK_RES_ERR_MAXNUMCON", 210
"MSK_RES_ERR_MAXNUMCONE", 213
"MSK_RES_ERR_MAXNUMQNZ", 210
"MSK_RES_ERR_MAXNUMVAR", 210
"MSK_RES_ERR_MIO_INTERNAL", 218
"MSK_RES_ERR_MIO_INVALID_NODE_OPTIMIZER", 220
"MSK_RES_ERR_MIO_INVALID_ROOT_OPTIMIZER", 220
"MSK_RES_ERR_MIO_NO_OPTIMIZER", 215
"MSK_RES_ERR_MISSING_LICENSE_FILE", 204
"MSK_RES_ERR_MIXED_CONIC_AND_NL", 215
"MSK_RES_ERR_MPS_CONE_OVERLAP", 207
"MSK_RES_ERR_MPS_CONE_REPEAT", 207
"MSK_RES_ERR_MPS_CONE_TYPE", 207
"MSK_RES_ERR_MPS_DUPLICATE_Q_ELEMENT", 207
"MSK_RES_ERR_MPS_FILE", 206
"MSK_RES_ERR_MPS_INV_BOUND_KEY", 206
"MSK_RES_ERR_MPS_INV_CON_KEY", 206
"MSK_RES_ERR_MPS_INV_FIELD", 206
"MSK_RES_ERR_MPS_INV_MARKER", 206
"MSK_RES_ERR_MPS_INV_SEC_NAME", 206
"MSK_RES_ERR_MPS_INV_SEC_ORDER", 207
"MSK_RES_ERR_MPS_INVALID_OBJ_NAME", 207
"MSK_RES_ERR_MPS_INVALID_OBJSENSE", 207
"MSK_RES_ERR_MPS_MUL_CON_NAME", 207
"MSK_RES_ERR_MPS_MUL_CSEC", 207
"MSK_RES_ERR_MPS_MUL_QOBJ", 207
"MSK_RES_ERR_MPS_MUL_QSEC", 207
"MSK_RES_ERR_MPS_NO_OBJECTIVE", 207
"MSK_RES_ERR_MPS_NON_SYMMETRIC_Q", 207
"MSK_RES_ERR_MPS_NULL_CON_NAME", 206
"MSK_RES_ERR_MPS_NULL_VAR_NAME", 206
"MSK_RES_ERR_MPS_SPLITTED_VAR", 207
"MSK_RES_ERR_MPS_TAB_IN_FIELD2", 207
"MSK_RES_ERR_MPS_TAB_IN_FIELD3", 207
"MSK_RES_ERR_MPS_TAB_IN_FIELD5", 207
"MSK_RES_ERR_MPS_UNDEF_CON_NAME", 206
"MSK_RES_ERR_MPS_UNDEF_VAR_NAME", 206
"MSK_RES_ERR_MUL_A_ELEMENT", 210
"MSK_RES_ERR_NAME_IS_NULL", 215
"MSK_RES_ERR_NAME_MAX_LEN", 215
"MSK_RES_ERR_NAN_IN_BLC", 214
"MSK_RES_ERR_NAN_IN_BLX", 214
"MSK_RES_ERR_NAN_IN_BUC", 214
"MSK_RES_ERR_NAN_IN_BUX", 214
"MSK_RES_ERR_NAN_IN_C", 214
"MSK_RES_ERR_NAN_IN_DOUBLE_DATA", 214
"MSK_RES_ERR_NEGATIVE_APPEND", 211
"MSK_RES_ERR_NEGATIVE_SURPLUS", 211
"MSK_RES_ERR_NEWER_DLL", 205
"MSK_RES_ERR_NO_BARS_FOR_SOLUTION", 217
"MSK_RES_ERR_NO_BARX_FOR_SOLUTION", 217
"MSK_RES_ERR_NO_BASIS_SOL", 215
"MSK_RES_ERR_NO_DUAL_FOR_ITG_SOL", 216
"MSK_RES_ERR_NO_DUAL_INFEAS_CER", 216
"MSK_RES_ERR_NO_INIT_ENV", 205
"MSK_RES_ERR_NO_OPTIMIZER_VAR_TYPE", 215
"MSK_RES_ERR_NO_PRIMAL_INFEAS_CER", 215
"MSK_RES_ERR_NO_SNX_FOR_BAS_SOL", 216
"MSK_RES_ERR_NO_SOLUTION_IN_CALLBACK", 216
"MSK_RES_ERR_NON_UNIQUE_ARRAY", 218
"MSK_RES_ERR_NONCONVEX", 212
"MSK_RES_ERR_NONLINEAR_EQUALITY", 212
"MSK_RES_ERR_NONLINEAR_FUNCTIONS_NOT_ALLOWED", 214
"MSK_RES_ERR_NONLINEAR_RANGED", 212
"MSK_RES_ERR_NR_ARGUMENTS", 209
"MSK_RES_ERR_NULL_ENV", 205
"MSK_RES_ERR_NULL_POINTER", 206
"MSK_RES_ERR_NULL_TASK", 205
"MSK_RES_ERR_NUMCONLIM", 210
"MSK_RES_ERR_NUMVARLIM", 210
"MSK_RES_ERR_OBJ_Q_NOT_NSD", 212
"MSK_RES_ERR_OBJ_Q_NOT_PSD", 212
"MSK_RES_ERR_OBJECTIVE_RANGE", 211
"MSK_RES_ERR_OLDER_DLL", 205
"MSK_RES_ERR_OPEN_DL", 205
"MSK_RES_ERR_OPF_FORMAT", 208
"MSK_RES_ERR_OPF_NEW_VARIABLE", 208
"MSK_RES_ERR_OPF_PREMATURE_EOF", 208
"MSK_RES_ERR_OPTIMIZER_LICENSE", 204
"MSK_RES_ERR_OVERFLOW", 215
"MSK_RES_ERR_PARAM_INDEX", 209
"MSK_RES_ERR_PARAM_IS_TOO_LARGE", 209
"MSK_RES_ERR_PARAM_IS_TOO_SMALL", 209
"MSK_RES_ERR_PARAM_NAME", 209
"MSK_RES_ERR_PARAM_NAME_DOU", 209
"MSK_RES_ERR_PARAM_NAME_INT", 209
"MSK_RES_ERR_PARAM_NAME_STR", 209
"MSK_RES_ERR_PARAM_TYPE", 209
"MSK_RES_ERR_PARAM_VALUE_STR", 209
"MSK_RES_ERR_PLATFORM_NOT_LICENSED", 204
"MSK_RES_ERR_POSTSOLVE", 215
"MSK_RES_ERR_PRO_ITEM", 212
"MSK_RES_ERR_PROB_LICENSE", 204
"MSK_RES_ERR_QCON_SUBI_TOO_LARGE", 213
"MSK_RES_ERR_QCON_SUBI_TOO_SMALL", 213

"MSK_RES_ERR_QCON_UPPER_TRIANGLE", 214
"MSK_RES_ERR_QOBJ_UPPER_TRIANGLE", 213
"MSK_RES_ERR_READ_FORMAT", 206
"MSK_RES_ERR_READ_LP_MISSING_END_TAG", 208
"MSK_RES_ERR_READ_LP_NONEXISTING_NAME", 208
"MSK_RES_ERR_REMOVE_CONE_VARIABLE", 213
"MSK_RES_ERR_REPAIR_INVALID_PROBLEM", 215
"MSK_RES_ERR_REPAIR_OPTIMIZATION_FAILED", 215
"MSK_RES_ERR_SEN_BOUND_INVALID_LO", 217
"MSK_RES_ERR_SEN_BOUND_INVALID_UP", 217
"MSK_RES_ERR_SEN_FORMAT", 216
"MSK_RES_ERR_SEN_INDEX_INVALID", 217
"MSK_RES_ERR_SEN_INDEX_RANGE", 217
"MSK_RES_ERR_SEN_INVALID_REGEXP", 217
"MSK_RES_ERR_SEN_NUMERICAL", 217
"MSK_RES_ERR_SEN_SOLUTION_STATUS", 217
"MSK_RES_ERR_SEN_UNDEF_NAME", 217
"MSK_RES_ERR_SEN_UNHANDLED_PROBLEM_TYPE", 217
"MSK_RES_ERR_SERVER_CONNECT", 220
"MSK_RES_ERR_SERVER_PROTOCOL", 220
"MSK_RES_ERR_SERVER_STATUS", 220
"MSK_RES_ERR_SERVER_TOKEN", 220
"MSK_RES_ERR_SIZE_LICENSE", 203
"MSK_RES_ERR_SIZE_LICENSE_CON", 204
"MSK_RES_ERR_SIZE_LICENSE_INTVAR", 204
"MSK_RES_ERR_SIZE_LICENSE_NUMCORES", 217
"MSK_RES_ERR_SIZE_LICENSE_VAR", 204
"MSK_RES_ERR_SOL_FILE_INVALID_NUMBER", 213
"MSK_RES_ERR_SOLITEM", 210
"MSK_RES_ERR_SOLVER_PROBTYPE", 211
"MSK_RES_ERR_SPACE", 205
"MSK_RES_ERR_SPACE_LEAKING", 206
"MSK_RES_ERR_SPACE_NO_INFO", 206
"MSK_RES_ERR_SYM_MAT_DUPLICATE", 218
"MSK_RES_ERR_SYM_MAT_HUGE", 214
"MSK_RES_ERR_SYM_MAT_INVALID", 214
"MSK_RES_ERR_SYM_MAT_INVALID_COL_INDEX", 218
"MSK_RES_ERR_SYM_MAT_INVALID_ROW_INDEX", 218
"MSK_RES_ERR_SYM_MAT_INVALID_VALUE", 218
"MSK_RES_ERR_SYM_MAT_NOT_LOWER_TRINGULAR", 218
"MSK_RES_ERR_TASK_INCOMPATIBLE", 216
"MSK_RES_ERR_TASK_INVALID", 216
"MSK_RES_ERR_TASK_WRITE", 216
"MSK_RES_ERR_THREAD_COND_INIT", 205
"MSK_RES_ERR_THREAD_CREATE", 205
"MSK_RES_ERR_THREAD_MUTEX_INIT", 205
"MSK_RES_ERR_THREAD_MUTEX_LOCK", 205
"MSK_RES_ERR_THREAD_MUTEX_UNLOCK", 205
"MSK_RES_ERR_TOCONIC_CONSTR_NOT_CONIC", 220
"MSK_RES_ERR_TOCONIC_CONSTR_Q_NOT_PSD", 220
"MSK_RES_ERR_TOCONIC_CONSTRAINT_FX", 220
"MSK_RES_ERR_TOCONIC_CONSTRAINT_RA", 220
"MSK_RES_ERR_TOCONIC_OBJECTIVE_NOT_PSD", 220
"MSK_RES_ERR_TOO_SMALL_MAX_NUM_NZ", 210
"MSK_RES_ERR_TOO_SMALL_MAXNUMANZ", 210
"MSK_RES_ERR_UNB_STEP_SIZE", 217
"MSK_RES_ERR_UNDEF_SOLUTION", 211
"MSK_RES_ERR_UNDEFINED_OBJECTIVE_SENSE", 214
"MSK_RES_ERR_UNHANDLED_SOLUTION_STATUS", 218
"MSK_RES_ERR_UNKNOWN", 205
"MSK_RES_ERR_UPPER_BOUND_IS_A_NAN", 213
"MSK_RES_ERR_UPPER_TRIANGLE", 218
"MSK_RES_ERR_USER_FUNC_RET", 214
"MSK_RES_ERR_USER_FUNC_RET_DATA", 214
"MSK_RES_ERR_USER_NLO_EVAL", 214
"MSK_RES_ERR_USER_NLO_EVAL_HESSUBI", 214
"MSK_RES_ERR_USER_NLO_EVAL_HESSUBJ", 214
"MSK_RES_ERR_USER_NLO_FUNC", 214
"MSK_RES_ERR_WHICHITEM_NOT_ALLOWED", 210
"MSK_RES_ERR_WHICHSOL", 210
"MSK_RES_ERR_WRITE_LP_FORMAT", 208
"MSK_RES_ERR_WRITE_LP_NON_UNIQUE_NAME", 208
"MSK_RES_ERR_WRITE_MPS_INVALID_NAME", 207
"MSK_RES_ERR_WRITE_OPF_INVALID_VAR_NAME", 208
"MSK_RES_ERR_WRITING_FILE", 208
"MSK_RES_ERR_XML_INVALID_PROBLEM_TYPE", 217
"MSK_RES_ERR_Y_IS_UNDEFINED", 214

## Structures

bara, 148
barc, 147
callback, 150
cones, 147
cprisen, 149
duasen, 150
info, 150
names, 147
prisen, 149
prob, 145
res, 146
solution, 148
solver_solutions, 148
symbcon, 150
vprisen, 149

# INDEX


## B

basis identification, 81
basis type
    sensitivity analysis, 111
bound
    constraint, 15, 121
    linear optimization, 15
    variable, 15, 121

## C

CBF format, 274
certificate
    dual, 123, 126, 127, 129
    primal, 123, 125, 127
complementarity, 122
cone
    dual, 125
    quadratic, 18, 124
    rotated quadratic, 18, 124
    semidefinite, 19, 126
conic optimization, 18, 124
    infeasibility, 125
    interior-point, 85
    termination criteria, 86
conic quadratic optimization, 18
Conic quadratic reformulation, 50
constraint
    bound, 15, 121
    linear optimization, 15
    matrix, 15, 121, 129
    quadratic, 128
constraints
    lower limit, 129
    upper limit, 129
convex interior-point
    optimizers, 89
cut, 91

## D

decision
    variables, 129
determinism, 78
dual
    certificate, 123, 126, 127, 129
    cone, 125
    feasible, 122
    infeasible, 122, 123, 126, 127, 129
    problem, 121, 125, 126
    variable, 122, 125
duality
    conic, 125
    gap, 122
    linear, 121
    semidefinite, 126
dualizer, 76

## E

eliminator, 76
example
    qo1, 21
    quadratic objective, 21

## F

feasible
    dual, 122
    primal, 79, 86, 121
    problem, 121
format
    CBF, 274
    json, 290
    LP, 248
    MPS, 253
    OPF, 265
    OSiL, 289
    sol, 297
    task, 289

## G

gap
    duality, 122

## H

hot-start, 83

## I

infeasibility, 123, 125, 127
    conic optimization, 125
    linear optimization, 123
    semidefinite, 127
infeasible, 98
    dual, 122, 123, 126, 127, 129
    primal, 79, 86, 121, 123, 125, 127

problem, 121, 123, 125, 127
infeasible problems, 98
installation, 6
path setup, 7
requirements, 6
troubleshooting, 6, 8
integer
optimizer, 90
variable, 25
integer feasible
solution, 92
integer optimization, 25, 90
cut, 91
delayed termination criteria, 92
initial solution, 26
objective bound, 91
optimality gap, 93
relaxation, 91
termination criteria, 92
tolerance, 92
integer optimizer
logging, 93
interior-point
conic optimization, 85
linear optimization, 78
logging, 82, 88
optimizer, 78, 85
termination criteria, 80, 86
interior-point optimizer, 89


## J


json format, 290


## L


linear constraint matrix, 15
linear dependency, 76
linear optimization, 14, 121
bound, 15
constraint, 15
infeasibility, 123
interior-point, 78
objective, 15
simplex, 83
termination criteria, 80, 83
variable, 15
linearity interval, 110
logging
integer optimizer, 93
interior-point, 82, 88
optimizer, 82, 84, 88
simplex, 84
lower limit
constraints, 129
variables, 130
LP format, 248


## M


matrix
constraint, 15, 121, 129
semidefinite, 19
symmetric, 19
MIP, *see* integer optimization
mixed-integer, *see* integer
mixed-integer optimization, *see* integer optimization
MPS format, 253
free, 265


## N


near-optimal
solution, 81, 88, 92
numerical issues
presolve, 76
scaling, 77
simplex, 84


## O


objective, 121
linear optimization, 15
objective bound, 91
objective vector, 129
OPF format, 265
optimal
solution, 122
optimality gap, 93
optimization
conic quadratic, 124
linear, 14, 121
semidefinite, 126
optimizer
determinism, 78
integer, 90
interior-point, 78, 85
logging, 82, 84, 88
parallelization, 77
selection, 76, 78
simplex, 83
optimizers
convex interior-point, 89
OSiL format, 289


## P


pair sensitivity analysis
optimal partition type, 111
parallelization, 77
parameter
simplex, 84
positive semidefinite, 21
posynomial, 68
presolve, 75
eliminator, 76
linear dependency check, 76
numerical issues, 76
primal
certificate, 123, 125, 127
feasible, 79, 86, 121

infeasible, 79, 86, 121, 123, 125, 127
problem, 121, 125, 126
solution, 121
primal-dual
problem, 78, 85
solution, 122
problem
dual, 121, 125, 126
feasible, 121
infeasible, 121, 123, 125, 127
primal, 121, 125, 126
primal-dual, 78, 85
unbounded, 123

Q

qo1
example, 21
quadratic
constraint, 128
quadratic cone, 18, 124
quadratic objective
example, 21
quadratic optimization, 128
quality
solution, 93

R

relaxation, 91
response handling, 26
rotated quadratic cone, 18, 124

S

scaling, 77
semidefinite
cone, 19, 126
infeasibility, 127
matrix, 19
variable, 19, 126
semidefinite optimization, 19, 126
sensitivity analysis, 109
basis type, 111
separable convex optimization, 37
shadow price, 110
simplex
linear optimization, 83
logging, 84
numerical issues, 84
optimizer, 83
parameter, 84
termination criteria, 83
sol format, 297
solution
file format, 297
integer feasible, 92
near-optimal, 81, 88, 92
optimal, 122
primal, 121
primal-dual, 122
quality, 93
solution summary, 32, 34
symmetric
matrix, 19

T

task format, 289
termination criteria
conic optimization, 86
delayed, 92
integer optimization, 92
interior-point, 80, 86
linear optimization, 80, 83
simplex, 83
tolerance, 81, 88, 92
thread, 77
tolerance
integer optimization, 92
termination criteria, 81, 88, 92
troubleshooting
installation, 6

U

unbounded
problem, 123
upper limit
constraints, 129
variables, 130

V

variable, 121
bound, 15, 121
dual, 122, 125
integer, 25
linear optimization, 15
semidefinite, 19, 126
variables
decision, 129
lower limit, 130
upper limit, 130