MOSEK MATLAB Toolbox

Release 8.0.0.81

MOSEK ApS

2017

1 Introduction **1**
1.1 Why the MATLAB Toolbox? . . . 1
1.2 License agreement . . . 1

2 Installation **3**
2.1 Supported MATLAB versions . . . 3
2.2 Installation . . . 3
2.3 MATLAB Setup . . . 3
2.4 Testing the toolbox . . . 4
2.5 Troubleshooting . . . 5

3 Guidelines **7**
3.1 The **MOSEK** integration with MATLAB . . . 7
3.2 Caveats Using the MATLAB Compiler . . . 8
3.3 The license system . . . 8

4 Basic Tutorials **9**
4.1 The Basics Tutorial . . . 10
4.2 Linear Optimization . . . 10
4.3 Conic Quadratic Optimization . . . 14
4.4 Semidefinite Optimization . . . 15
4.5 Quadratic Optimization . . . 17
4.6 Integer Optimization . . . 21
4.7 Optimizer Termination Handling . . . 23
4.8 Problem Modification and Reoptimization . . . 24
4.9 Solution Analysis . . . 27
4.10 Solver Parameters . . . 31

5 Nonlinear Tutorials **33**
5.1 Separable Convex (SCopt) Interface . . . 33
5.2 Entropy Optimization . . . 36
5.3 Geometric Optimization . . . 37

6 Advanced Tutorials **41**
6.1 Linear Least Squares and Related Norm Minimization Problems . . . 41
6.2 Converting a quadratically constrained problem to conic form . . . 48

7 Case Studies **53**
7.1 Robust linear Optimization . . . 53
7.2 Geometric (posynomial) Optimization . . . 65

8 Managing I/O **71**
8.1 Stream I/O . . . 71
8.2 File I/O . . . 71
8.3 Verbosity . . . 72

9 The Optimizers for Continuous Problems **73**
9.1 Presolve . . . 73
9.2 Linear Optimization . . . 75
9.3 Conic Optimization . . . 81
9.4 Nonlinear Convex Optimization . . . 82
9.5 Using Multiple Threads in an Optimizer . . . 83

10 The Optimizer for Mixed-integer Problems **85**
10.1 Some Concepts and Facts Related to Mixed-integer Optimization . . . 85
10.2 The Mixed-integer Optimizer . . . 86
10.3 Termination Criterion . . . 86
10.4 Parameters Affecting the Termination of the Integer Optimizer. . . . 87
10.5 How to Speed Up the Solution Process . . . 87
10.6 Understanding Solution Quality . . . 88

11 Problem Analyzer **89**
11.1 General Characteristics . . . 90
11.2 Objective . . . 91
11.3 Linear Constraints . . . 91
11.4 Constraint and Variable Bounds . . . 92
11.5 Quadratic Constraints . . . 92
11.6 Conic Constraints . . . 92

12 Analyzing Infeasible Problems **93**
12.1 Example: Primal Infeasibility . . . 93
12.2 Locating the cause of Primal Infeasibility . . . 94
12.3 Locating the Cause of Dual Infeasibility . . . 95
12.4 The Infeasibility Report . . . 95
12.5 Theory Concerning Infeasible Problems . . . 97
12.6 The Certificate of Primal Infeasibility . . . 97
12.7 The certificate of dual infeasibility . . . 98

13 Sensitivity Analysis **103**
13.1 Sensitivity Analysis for Linear Problems . . . 103
13.2 Sensitivity Analysis with **MOSEK** . . . 109

14 Problem Formulation and Solutions **115**
14.1 Linear Optimization . . . 115
14.2 Conic Quadratic Optimization . . . 118
14.3 Semidefinite Optimization . . . 120
14.4 Quadratic and Quadratically Constrained Optimization . . . 122
14.5 General Convex Optimization . . . 123

15 Toolbox Reference **125**
15.1 Command Reference . . . 125
15.2 Data Structures and Notation . . . 136
15.3 Parameters . . . 144
15.4 Response codes . . . 192
15.5 Enumerations . . . 213

16 Supported File Formats **241**
16.1 The LP File Format . . . 242
16.2 The MPS File Format . . . 247
16.3 The OPF Format . . . 259
16.4 The CBF Format . . . 268
16.5 The XML (OSiL) Format . . . 283
16.6 The Task Format . . . 283
16.7 The JSON Format . . . 283
16.8 The Solution File Format . . . 290

17 Interface changes **293**
17.1 Compatibility 293
17.2 Parameters 293
17.3 Constants 295
17.4 Response Codes 299

Bibliography **301**

API Index **303**

CHAPTER

ONE

INTRODUCTION

The **MOSEK** Optimization Suite 8.0.0.81 is a powerfull software package capable of solving large-scale optimization problems of the following kind:

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

1.1 Why the MATLAB Toolbox?

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

1.2 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/sales/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 1.1.

Listing 1.1: 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 1.2.

Listing 1.2: 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

TWO

INSTALLATION

2.1 Supported MATLAB versions

The **MOSEK** optimization toolbox for MATLAB supports recent MATLAB versions as shown in Table 2.1.

Table 2.1: Supported MATLAB versions.

Platform	R2014a or newer
linux64x86	Yes
osx64x86	Yes
win32x86	Yes
win64x86	Yes

2.2 Installation

The installation procedure involves the following steps:

1. Install **MOSEK** Optimization Suite for the relevant platform and unpacking
2. MATLAB setup
3. Testing the installation.

Instructions for installing **MOSEK** Optimization Suite are in located in the Installation Guide.

2.3 MATLAB Setup

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

2.3.1 Windows

If you are using Windows you should do

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

or alternatively

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

if you do not want access to the overloaded MATLAB optimization toolbox functions such as *linprog* and *quadprog*.

2.3.2 Linux

If you are using Linux you should do

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

or alternatively

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

if you do not want access to the overloaded MATLAB optimization toolbox functions such as *linprog* and *quadprog*.

2.3.3 Mac OS

It is import on Mac OS to run the script `<MSKHOME>/mosek/8/tools/platform/osx64x86/bin/install.py` which make important updates to the installtion. Next inside MATLAB you should do

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

or alternatively

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

if you do not want access to the overloaded MATLAB optimization toolbox functions such as *linprog* and *quadprog*.

2.3.4 Permanently Changing `matlabpath`

Instead of running the `addpath` command every time MATLAB is started then the path be add to the startup file i.e. {matlab}\toolbox\local\startup.m‘, where matlab is the MATLAB root directory. Alternatively the permanent modification of the MATLAB path can be performed using the FileSet Path menu item.

2.4 Testing the toolbox

You can verify that **MOSEK** 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: Warnings about Fusion and the command line is not relevant if you only access **MOSEK** using this toolbox!

2.5 Troubleshooting

2.5.1 Undefined Function or Variable *mosekopt*

If you get the MATLAB error message

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

you have not set up the `matlabpath` correctly as described in Section 2.3.

2.5.2 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 from *mosekopt*:

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

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

2.5.3 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 versions of **MOSEK** and MATLAB is likely. I.e. the MATLAB 32 bit and **MOSEK** is 64 bit. 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 a MATLAB valid `.m` file,

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

CHAPTER

THREE

GUIDELINES

3.1 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.

3.1.1 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.

3.1.2 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 Section 15.1. 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.

3.1.3 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.

CHAPTER

FOUR

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.
- Semide nite 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.

4.1 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 4.1: 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
```

4.2 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

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

where we have used the problem elements:

- m and n which are the number of constraints and variables respectively,
- x which is the variable vector of length n,
- c which is a coefficient vector of size n

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

- c^f which is a constant,
- A which is a $m \times n$ matrix of coefficients is given by

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

- l^c and u^c which specify the lower and upper bounds on constraints respectively, and
- l^x and u^x which specifies the lower and upper bounds on variables respectively.

Note: Please note the unconventional notation using 0 as the first index rather than 1. Hence, x_0 is the first element in variable vector x.

4.2.1 Example LO1

The following is an example of a linear optimization problem:

$$\begin{array}{lrcrcrcrcr} \text{maximize} & 3x_0 & + & 1x_1 & + & 5x_2 & + & 1x_3 & & \\ \text{subject to} & 3x_0 & + & 1x_1 & + & 2x_2 & & & = & 30, \\ & 2x_0 & + & 1x_1 & + & 3x_2 & + & 1x_3 & \geq & 15, \\ & & & 2x_1 & & & + & 3x_3 & \leq & 25, \end{array} \tag{4.1}$$

having the bounds

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

Example: Linear optimization using *msklpopt*

A linear optimization problem such as (4.1) can be solved using the *msklpopt* function. The first step in solving the example (4.1) is to setup the data for problem (4.1) i.e. the c, A, etc. Afterwards the problem is solved using an appropriate call to *msklpopt*.

Listing 4.2: Script implementing problem (4.1).

```
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);
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 (4.1) and solve the problem using *mosekopt*.

Listing 4.3: Script implementing problem (4.1) 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('minimize',prob);

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

Please note that

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

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 Section 3.1 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.

4.3 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. Actually, since the set $\mathbb{R}^n$ of real numbers is also a convex cone, all variables can in fact be partitioned into subsets belonging to separate convex cones, simply stated $x \in \mathcal{K}$.

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, the user only specify subsets of variables x^t belonging to cones $\mathcal{K}_t$ different from the set $\mathbb{R}^{n_t}$ of real numbers. These cones can be a:

- 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:

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

From these definition it follows that

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

is equivalent to

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

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.

4.3.1 Example CQO1

We want to solve the following Conic Optimization Problem problem:

$$\begin{array}{lccc} \text{minimize} & x_4 + x_5 + x_6 & & \\ \text{subject to} & x_1 + x_2 + 2x_3 & = & 1, \\ & x_1, x_2, x_3 & \geq & 0, \\ & x_4 \geq \sqrt{x_1^2 + x_2^2}, & & \\ & 2x_5 x_6 \geq x_3^2 & & \end{array} \tag{4.2}$$

is an example of a conic quadratic optimization problem. The problem involves some linear constraints, a quadratic cone and a rotated quadratic cone.

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 4.4 demonstrates how to solve the example (4.2) using **MOSEK**.

Listing 4.4: Script implementing problem (4.2).

```
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 partiucular 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.

4.4 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

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

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

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

4.4.1 Example SDO1

The problem

$$\begin{array}{lcccl}
\text{minimize} & \left\langle \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}, \overline{X} \right\rangle + x_0 & & & \\
\text{subject to} & \left\langle \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \overline{X} \right\rangle + x_0 & = & 1, & \\
 & \left\langle \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \overline{X} \right\rangle + x_1 + x_2 & = & 1/2, & \\
 & x_0 \geq \sqrt{{x_1}^2 + {x_2}^2}, & & \overline{X} \succeq 0, &
\end{array} \tag{4.3}$$

is a mixed semidefinite and conic quadratic programming problem with a 3-dimensional semidefinite variable

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

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

$$\overline{X}_{00} + \overline{X}_{11} + \overline{X}_{22} + x_0 = 1,$$

and

$$\overline{X}_{00} + \overline{X}_{11} + \overline{X}_{22} + 2(\overline{X}_{10} + \overline{X}_{20} + \overline{X}_{21}) + x_1 + x_2 = 1/2.$$

Listing 4.5 demonstrates how to solve this problem using **MOSEK**.

Listing 4.5: Code implementing problem (4.3).

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

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

prob.bardim    = [3];
prob.barc.subj = [1, 1, 1, 1, 1];
```

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 the smallest eigenvalue of the matrix is nonnegative. An alternative statement of the positive semidefinite requirement is

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

If Q is not positive semidefinite, then **MOSEK** will not produce reliable results or work at all.

One way of checking whether Q is positive semidefinite is to check whether all the eigenvalues of Q are nonnegative.

4.5.1 Example: Quadratic Objective

$$\begin{array}{lll} \text{minimize} & & x_1^2 + 0.1x_2^2 + x_3^2 - x_1x_3 - x_2 \\ \text{subject to} & 1 \leq & x_1 + x_2 + x_3 \\ & & x \geq 0. \end{array} \tag{4.6}$$

For the example (4.6) implies that

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

and that

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

Please note the explicit $\frac{1}{2}$ in the objective function of (4.4) which implies that diagonal elements must be doubled in Q, i.e. $Q_{11} = 2$, whereas the coefficient in (4.6) is 1 in front of x_1^2.

Important: **MOSEK** assumes that the Q matrix is symmetric, i.e. $Q = Q^T$, and that Q is positive semide nite.

Using `mosekopt`

In Listing 4.6 we show how to use *`mosekopt`* to solve problem (4.6). This is the preferred way.

Listing 4.6: How to solve problem (4.6) 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 4.7 we show how to use *mskqpopt* to solve problem (4.6).

Listing 4.7: Function solving problem (4.6) 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.

4.5.2 Example: Quadratic constraints

In this section describes how to solve a problem with quadratic constraints. Please note that quadratic constraints are subject to the convexity requirement (4.5).

Consider the problem:

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

This is equivalent to

$$\begin{array}{lll} \text{minimize} & \frac{1}{2}x^T Q^o x + c^T x & \\ \text{subject to} & \frac{1}{2}x^T Q^0 x + Ax & \geq \quad b, \end{array} \tag{4.7}$$

where

$$Q^o = \begin{bmatrix} 2 & 0 & -1 \\ 0 & 0.2 & 0 \\ -1 & 0 & 2 \end{bmatrix}, c = \begin{bmatrix} 0 & -10 \end{bmatrix}, A = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, b = 1.$$

$$Q^0 = \begin{bmatrix} -2 & 0 & 0.2 \\ 0 & -2 & 0 \\ 0.2 & 0 & -0.2 \end{bmatrix}.$$

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

Listing 4.8: Script implementing problem (4.7).

```
function qcqo1()
clear prob;

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

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

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

prob.buc    = [1 0.5]';

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

% Display the solution.
fprintf('\nx:');
```

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.

Solving a mixed-integer optimization program could easily result in long running time. It is therefore of interest to consider a termination criterion based on the maximum running time. This is possible setting the *MSK_DPAR_MIO_MAX_TIME*. See Section 4.10 for more details on how to set solver parameters.

4.6.2 Specifying an initial solution

Integer optimization problems are generally hard to solve, but the solution time 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, will force **MOSEK** to compute the remaining continuous variable values.

If the specified integer solution is infeasible or incomplete, **MOSEK** will simply ignore it.

Consider the problem

$$\begin{array}{ll} \text{maximize} & 7x_0 + 10x_1 + x_2 + 5x_3 \\ \text{subject to} & x_0 + x_1 + x_2 + x_3 \leq 2.5 \\ & x_0, x_1, x_2 \in \mathbb{Z} \\ x_0, x_1, x_2, x_3 \geq 0 & \end{array} \tag{4.9}$$

The following example demonstrates how to optimize the problem using a feasible starting solution generated by selecting the integer values as $x_0 = 0, x_1 = 2, x_2 = 0$.

Listing 4.10: Script solving problem (4.9).

```
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
```

4.7 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 *MSKrescodetypee*. See 15.4 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 4.11 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 4.11: 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
```

4.8 Problem Modification and Reoptimization

Often one might want to solve not just a single optimization problem, but a sequence of problem, 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 three categories:

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

These operations may be costly and, especially removing variables and constraints. 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.

For instance the former optimal solution may be still feasibile, but no more optimal; or for tiny modifications of the objective function it may be still optimal. This is a special case that we discuss in Section 13.

In general, **MOSEK** exploits dual information and the availablity 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 reliable and effective as those for the simplex algorithm. More information can be found in Chapter 10 of the book [Chv83].

4.8.1 Example: Production Planning

A company manufactures three types of products. Suppose the stages of manufacturing can be split into three parts, namely 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.

Now the question is 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 the linear optimization problem:

$$\begin{array}{lrcrcrcl} \text{maximize} & 1.5x_0 & + & 2.5x_1 & + & 3.0x_2 & & \\ \text{subject to} & 2x_0 & + & 4x_1 & + & 3x_2 & \leq & 100000, \\ & 3x_0 & + & 2x_1 & + & 3x_2 & \leq & 50000, \\ & 2x_0 & + & 3x_1 & + & 2x_2 & \leq & 60000, \end{array} \tag{4.10}$$

and

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

Code in Listing 4.12 loads and solves this problem:

Listing 4.12: How to load problem (4.10)

```
% 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
```

4.8.2 Changing the A 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:

$$\begin{array}{lrcrcrcl} \text{maximize} & 1.5x_0 & + & 2.5x_1 & + & 3.0x_2 & & \\ \text{subject to} & 3x_0 & + & 4x_1 & + & 3x_2 & \leq & 100000, \\ & 3x_0 & + & 2x_1 & + & 3x_2 & \leq & 50000, \\ & 2x_0 & + & 3x_1 & + & 2x_2 & \leq & 60000, \end{array} \tag{4.11}$$

and

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

After changing the A matrix we can find the new optimal solution by calling *mosekopt* again

4.8.3 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 value in the objective. We do this in Listing 4.13

Listing 4.13: How to add a 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 problem looks this way:

$$\begin{array}{lrcrcrcrcl} \text{maximize} & 1.5x_0 & + & 2.5x_1 & + & 3.0x_2 & + & 1.0x_3 & & \\ \text{subject to} & 3x_0 & + & 4x_1 & + & 3x_2 & + & 4x_3 & \leq & 100000, \\ & 3x_0 & + & 2x_1 & + & 3x_2 & & & \leq & 50000, \\ & 2x_0 & + & 3x_1 & + & 2x_2 & + & 1x_3 & \leq & 60000, \end{array} \tag{4.12}$$

and

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

4.8.4 Reoptimization

When *mosekopt* is called **MOSEK** will store the optimal solution internally. After a the problem has been modified and *mosekopt* is called again the solution will automatically be used to reduce solution time of the new problem, if possible.

In this case an optimal solution to problem (4.11) was found and then added a column was added to get (4.12). We let **MOSEK** select the suitable simplex algorithm to perform reoptimization.

```
% select the primal simplex
param.MSK_IPAR_OPTIMIZER = 'MSK_OPTIMIZER_FREE_SIMPLEX';

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

4.8.5 Appending Constraints

Now suppose we want to add a new stage to the production 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 which is done in the following code:

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

4.9 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 Section 4.9.1). 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 Section 11.

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 Section 1.2).

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 le 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**.

4.9.1 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.

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

Listing 4.14: 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
```

4.9.2 The Solution Summary for Mixed-Integer Problems

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

Listing 4.15: 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.

4.10 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 Section 15.3.

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 *MSKoptimizertypee* 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 Section 15.1.

The complete code for this tutorial follows in Listing 4.16.

Listing 4.16: 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

FIVE

NONLINEAR TUTORIALS

This chapter provides information about how to solve general convex nonlinear optimization problems using **MOSEK**. By general nonlinear problems it is meant problems that cannot be formulated as a conic quadratic optimization or a convex quadratically constrained optimization problem.

In general it is recommended not to use nonlinear optimizer unless needed. The reasons are

- **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 hard to provide correctly i.e. it is nontrivial to program the code that computes the derivative information.
- The algorithm employed for nonlinear optimization problems is not as good as the one employed for conic problems i.e. conic problems has special that can be exploited to make the optimizer faster and more robust.

This leads to following advices in decreasing order of importance.

1. 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.
2. 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 on separable form also solves faster because of the simpler structure of the functions.
3. Finally, if the problem cannot be reformulated to separable form then use a modelling language like AMPL or GAMS. The reason is the modeling language will do all the computing of 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 following tutorials are provided in this section:

- The Separable Convex interface (SCopt)
- Entropy optimization
- Geometric optimization

5.1 Separable Convex (SCopt) Interface

The **MOSEK** toolbox API 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 a normal linear problem with additional non-linear terms z:

$$\begin{array}{lcccccc} \text{minimize} & & & z_0(x) + c^T x & & \\ \text{subject to} & l_i^c & \leq & z_i(x) + a_i^T x & \leq & u_i^c, \ i = 1 \ldots m \\ & l^x & \leq & x & \leq & u^x, \\ & x \in \mathbb{R}^n & z : \mathbb{R}^n \to \mathbb{R}^{(m+1)} & & & \end{array}$$

Using the separable non-linear interface it is possible to add non-linear functions of the form

$$z_i(x) = \sum_{k=1}^{K_i} w_k^i(x_{p_{ik}}),\ w_k^i : \mathbb{R} \to \mathbb{R}$$

In other words, each non-linear function z_i is a sum of separable functions w_k^i of one variable each. A limited set of functions are supported; each w_k^i can be one of the separable functions:

Table 5.1: Functions supported by the SCopt interface.

Separable function	Operator 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 and h are constants.

This formulation does not guarantee convexity. For **MOSEK** to be able to solve the problem, 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.

If these are not satisfied **MOSEK** may not be able to solve the problem or produce a meaningful status report. For details see Section 5.1.1.

Important: When to use the SCopt API:

- When conic can absolutely not be used.
- when a conic formulation would be significantly larger

Problems: less stable, less predictable, harder to debug, worse status info

---

5.1.1 Ensuring Convexity and Differentiability

Some simple rules can be set up 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.

We can define these bounds as follows:

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. The following table lists convexity for the relevant ranges for $f > 0$ — changing the sign of f switches concavity/convexity.

Separable function	Operator name	Convexity conditions
$f x \ln(x)$	ent	Convex within safe bounds.
$f e^{gx+h}$	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: ˆ concave if $(1 \quad ; \quad h)$, ˆ convex if $(\quad h; 1)$ If $0 < g < 1$: concave within safe bounds.
		Otherwise: convex within safe bounds.

5.1.2 SCopt Example

Subsequently, we will use the following example to demonstrate the solution of a separable convex optimization problem using MOSEK

$$\begin{array}{lll} \text{maximize} & \exp(x_2) & \ln(x_1) \\ \text{subject to} & x_2 \ln(x_2) & 0 \\ & \sqrt{x_1^2 \quad x_2} & 0 \\ & x_1; x_2 & \frac{1}{2}: \end{array} \tag{5.1}$$

This problem is obviously separable. Moreover, note that all nonlinear functions are well de ned for x values satisfying the variable bounds strictly, i.e.

$$x_1; x_2 > 0:$$

This assures that function evaluation errors will not occur during the optimization process because MOSEK will only evaluate $\ln(x_1)$ and $x_2 \ln(x_2)$ for $x_1; x_2 > 0$.

The method employed above can often be used to make convex optimization problems separable even if these are not formulated as such initially. The reader might object that this approach is ine cient because additional constraints and variables are introduced to make the problem separable. However, in our experience this drawback is o set largely by the much simpler structure of the nonlinear functions. Particularly, the evaluation of the nonlinear functions, their gradients and Hessians is much easier in the separable case.

The MOSEK MATLAB Toolbox 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 (5.1), the linear part of the problem, such as c and A, is speci ed as usual using MATLAB vectors and matrices. However, the nonlinear functions must be speci ed using ve arrays which in the case of problem (5.1) 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,

- ˆ opr(k,:) speci es the type of a nonlinear function,
- ˆ opri(k) speci es in which constraint the nonlinear function should be added (zero means objective),
- ˆ 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 5.1.

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.

The complete source code follows in Listing 5.1.

Listing 5.1: Implementation of problem (5.1).

```
function scopt1()
% Specify the linear part of the problem.

c           = [0;0];
a           = sparse([[0 0];[0 -1 ]]);
blc         = [-inf;    0];
buc         = [   0;  inf];
blx         = [ 0.5;  0.5];
bux         = [ 1.0;  1.0];

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];


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

[res]       = mskscopt(opr,opri,oprj,oprf,oprg,c,a,blc,buc,blx,bux);


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

5.2 Entropy Optimization

An entropy optimization problem has the following form

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

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

Example

An example of an entropy optimization problem is

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

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

Listing 5.2: 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
```

5.3 Geometric Optimization

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

$$\begin{array}{lll} \text{minimize} & \sum_{k\in J_0} c_k \prod_{j=1}^n t_j^{a_{kj}} & \\ \text{subject to} & \sum_{k\in J_i} c_k \prod_{j=1}^n t_j^{a_{kj}} & \leq \quad 1, \quad i=1,\ldots,m, \\ & t>0, & \end{array} \tag{5.2}$$

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 (5.2) is very hard to solve, but the posynomial case where

$$c > 0$$

is relatively easy. Using the variable transformation

$$t_j = e^{x_j} \tag{5.3}$$

we obtain the problem

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

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

$$\begin{array}{lll} \text{minimize} & \log(\sum_{k\in J_0} c_k e^{a_{k:}x}) & \\ \text{subject to} & \log(\sum_{k\in J_i} c_k e^{a_{k:}x}) & \leq \quad \log(1), \quad i=1,\ldots,m, \end{array} \tag{5.4}$$

which is also a convex optimization problem since log is strictly increasing. Hence, the problem (5.4) 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 (5.4) 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 (5.3).

Subsequently, we will use the example

$$\begin{array}{lll} \text{minimize} & 40t_1^{-1}t_2^{-\frac{1}{2}}t_3^{-1} + 20t_1t_3 + 40t_1t_2t_3 & \\ \text{subject to} & \frac{1}{3}t_1^{-2}t_2^{-2} + \frac{4}{3}t_2^{\frac{1}{2}}t_3^{-1} & \leq \quad 1, \\ & 0 < t_1, t_2, t_3 & \end{array} \tag{5.5}$$

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

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

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 map_k equals zero, the kth term belongs to the objective function, otherwise it belongs to the map_knth constraint.

Listing 5.3 demonstrates how the example is solved using *mskgpopt*.

Listing 5.3: Example on how to use mskgpopt.

```
%%
%
%  Copyright : $$copyright
%
%  File :      $${file}
%
%%

%%TAG:begin-code
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);
```

CHAPTER

SIX

ADVANCED TUTORIALS

6.1 Linear Least Squares and Related Norm Minimization Problems

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

$$\text{minimize} \|Fx - b\| \tag{6.1}$$

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.

6.1.1 The Case of the 2-norm

Initially let us focus on the 2 norm. In this case (6.1) is identical to the quadratic optimization problem

$$\text{minimize } \frac{1}{2}x^T F^T F x + \frac{1}{2}b^T b - b^T F x \tag{6.2}$$

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 (6.2) is solved using *mosekopt*. In the example the problem data is read from a file, then data for the problem (6.2) is constructed and finally the problem is solved.

Listing 6.1: Script solving problem (6.2)

```
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.
```

and hence at optimum

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

holds. Problem (6.3) is straightforward to solve, for instance using script as in Listing 6.3

Listing 6.3: Script solving problem (6.3).

```
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)
```

6.1.3 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

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

which in turn is equivalent to

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

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

$$\begin{array}{lll} \text{minimize} & e^T t & \\ \text{subject to} & Fx - te & \leq \quad b, \\ & Fx + te & \geq \quad b, \end{array} \tag{6.4}$$

where $e = (1, \ldots, 1)^T$. Next, this problem is solved in Listing 6.4.

Listing 6.4: Script solving problem (6.4).

```
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 (6.3). Indeed problem (6.3) is equivalent to

$$\begin{array}{lcccl} \text{minimize} & \sum_{i=1}^m t_i & & & \\ \text{subject to} & f_{i:}x - b_i - t_i + v_i & = & 0, & i = 1, \ldots, m, \\ & -(f_{i:}x - b_i) - t_i & \leq & 0, & i = 1, \ldots, m, \\ & v_i \geq 0, & & & i = 1, \ldots, m. \end{array} \tag{6.5}$$

After eliminating the t variables then this problem is equivalent to

$$\begin{array}{lcccl} \text{minimize} & \sum_{i=1}^m (f_{i:}x - b_i + v_i) & & & \\ \text{subject to} & -2(f_{i:}x - b_i) - v_i & \leq & 0, & i = 1, \ldots, m, \\ & v_i \geq 0, & & & i = 1, \ldots, m. \end{array} \tag{6.6}$$

Please note that this problem has only half the number of general constraints than problem (6.3) 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 (6.6) is likely to require much less space than the problem (6.5).

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

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|Fx - f\|_2^2 \\ \text{subject to} & Ax = b, \\ & l^x \leq x \leq u^x, \end{array} \tag{6.7}$$

where F and A are matrices and the remaining quantities are vectors. x is the vector of decision variables. The problem (6.7) 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

$$\begin{array}{lcll} \text{minimize} & \frac{1}{2}\|z\|_2^2 & & \\ \text{subject to} & Ax & = & b, \\ & Fx - z & = & f, \\ & l^x \leq x \leq u^x. & & \end{array} \tag{6.8}$$

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 (6.8) is much sparser than in problem (6.7). 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 (6.8) is not attractive whereas the corresponding dual problem is. Using the duality theory outlined in Section 14.5.1 we obtain the dual problem

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

which can be simplified to

$$\begin{array}{ll} \text{maximize} & b^T y + f^T z + (l^x)^T s_l^x + (u^x)^T s_u^x - \frac{1}{2}\|z\|_2^2 \\ \text{subject to} & A^T y + F^T z + s_l^x - s_u^x = 0, \\ & s_l^x, s_u^x \geq 0 \end{array} \tag{6.9}$$

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 (6.9) will have much fewer constraints than the primal problem (6.8); in general, the fewer constraints a problem contains, the more efficient **MOSEK** tends to be. A question is: If the dual problem (6.9) is solved instead of the primal problem (6.8), 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 6.5: 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 6.6: 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.

6.1.4 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

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

is equivalent to

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

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

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

which can be reduced to

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

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.

6.2 Converting a quadratically constrained problem to conic form

A conic quadratic constraint has the form

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

in its most basic form where

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

Alternatively the conic constraint can be represented using a quadratic inequality

$$\sum_{j=2}^{n} x_j^2 - x_1^2 \leq 0.0 \tag{6.10}$$

and the simple linear inequality

$$x_1 \geq 0.0.$$

Therefore, it is possible to state conic quadratic problems using quadratic inequalities. Some drawbacks of specifying conic quadratic problems using quadratic inequalities are

- the elegant duality theory for conic problems is lost.
- reporting accurate dual information for quadratic inequalities is hard and/or compuational expensive.
- the left hand side of (6.10) is nonconvex so the formulation is stricly speaking not convex.
- a computational overhead is introduced when converting the quadratic inequalities to conic form before optimizing.
- modelling directly on conic form usually leads to a better model [And13] i.e. a faster solution time and better numerical properties.

In addition quadratic inequalities can not be used to speciffy the semidefinite cone or other more general cones than quadratic cone. Despite the drawbacks it is not uncommon to state conic quadratic problems

using quadratic inequalities and therefore **MOSEK** has a function that translate certain quadratically constrained problems to conic form. Note that the **MOSEK** interior-point optimizer will do that automatically for convex quadratic problems automatically. So quadratic to conic form conversion is primarily useful for problems having conic quadratic constraints embedded.

MOSEK employs the following form of quadratic problems:

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

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,
- for each reformulated quadratic constraint there will be:
  - one rotated quadratic cone for each quadratic constraint,
  - one rotated quadratic cone if the objective function is quadratic,
  - 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.

Note: `mosekopt('toconic')` modifies the input task in-place: this means that if the reformulation is not possible, i.e. the problem is not conic representable, the state of the task is in general undefined.

6.2.1 Quadratic Constraint Reformulation

Let us assume we want to convert the following quadratic constraint

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

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

$$\frac{1}{2}x^T Q x + \sum_{j=0}^{n-1} a_j^T x_j \leq u. \tag{6.11}$$

Introducing an additional variable w such that

$$w = u - \sum_{j=0}^{n-1} a_j^T x_j \tag{6.12}$$

we obtain the equivalent form

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

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

$$Q = FF^T \tag{6.13}$$

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

$$\begin{array}{ll} (w, z, y) & \in \mathcal{Q}_r, \\ z = 1. & \\ y = Fx & \\ w = u - \sum_{j=0}^{n-1} a_j^T x_j. & \end{array} \tag{6.14}$$

Summarizing, for each quadratic constraint involving $t \leq n$ 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.

6.2.2 Some Examples

We report in this section few examples of reformulation of a QCQP problem in conic form. For each problem we will show its definition before and after the reformulation, using the human-readable OPF format.

Quadratic problem

We consider a simple quadratic problem of the form

$$\begin{array}{lll} \min & \frac{1}{2}(13x_0^2 + 17x_1^2 + 12x_2^2 + 24x_0x_1 + 12x_1x_2 - 4x_0x_2) - 22x_0 - 14.5x_1 + 12x_2 + 1 & \\ s.t. & & \\ & -1 \leq x_i \leq 1 & i = 0, 1, 2 \end{array}$$

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


[variables disallow_new_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 solution is attained for $x^\star = (1, 0.5, -1)$. The conversion will introduce first a variable x_3 in the objective function such that $x_3 \geq 1/2x^TQx$ and then convert the

latter directly in conic form. The converted problem follows:

$$
\begin{array}{ll}
\min & 22x_0 \quad 14.5ex_1 + 12x_2 + x_3 + 1 \\
\text{s.t.} & \\
& 3.61x_0 + 3.33x_1 \quad 0.55x_2 \quad x_6 = 0 \\
& \quad +2.29x_1 3.42x_2 \quad x_7 = 0 \\
& \quad 0.81x_1 \quad x_8 = 0 \\
& \quad x_3 + x_4 = 0 \\
& \quad x_5 = 1 \\
& \quad (x_4; x_5; x_6; x_7; x_8) \in Q_r \\
& \quad 1 \quad x_0; x_1; x_2 \quad 1
\end{array}
$$

The model generated bymosekopt(' write(prob.opf) ') is

```
[comment]
Written by MOSEK version 8.0.0.8
Date 25-05-15
Time 15:51:41
[/comment]


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

We can clearly see that constraints c0000 to c0002 represent the linear mapping as in (6.13), while (6.12) corresponds to c0003. The cone roots are x0005 and x0004.

CHAPTER

SEVEN

CASE STUDIES

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

Case Studies	Type	Int.	Keywords
Robust linear optimization	CQO	NO	Robust optimization
Geometric optimization	EXPOPT	NO	Posynomial optimization

7.1 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] .

7.1.1 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

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

and maximize the income

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

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

Minimize

$$-\{100 \cdot \texttt{RawI} + 199.90 \cdot \texttt{RawII} + 700 \cdot \texttt{DrugI} + 800 \cdot \texttt{DrugII}\} + \{6200 \cdot \texttt{DrugI} + 6900 \cdot \texttt{DrugII}\} \quad (7.1)$$

subject to

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

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 (7.1) 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 7.1 is the MATLAB script which specifies the problem and solves it using the **MOSEK** optimization toolbox:

Listing 7.1: 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 7.2: 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.

7.1.2 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^{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}^{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

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

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

7.1.3 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

$$\begin{array}{lrcccl} \text{minimize} & & & c^T x & & \\ \text{subject to} & l_c & \leq & Ax & \leq & u_c, \\ & l_x & \leq & x & \leq & u_x, \end{array} \tag{7.2}$$

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:

$$\begin{array}{rcl} \mathcal{U} & = & \{(c, A, l_c, u_c, l_x, u_x) : \\ & & (c^{\mathrm{n}} - dc, A^{\mathrm{n}} - dA, l_c^{\mathrm{n}} - dl_c, u_c^{\mathrm{n}} - du_c, l_x^{\mathrm{n}} - dl_x, u_x^{\mathrm{n}} - du_x) \leq (c, A, l_c, u_c, l_x, u_x) \\ & & \leq (c^{\mathrm{n}} + dc, A^{\mathrm{n}} + dA, l_c^{\mathrm{n}} + dl_c, u_c^{\mathrm{n}} + du_c, l_x^{\mathrm{n}} + dl_x, u_x^{\mathrm{n}} + du_x)\}. \end{array} \tag{7.3}$$

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 (7.2) with data running through a given uncertainty set $\mathcal{U}$ is called an uncertain linear optimization problem.
- Vector x is called a robust feasible solutionto 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 objectiveat x does not exceed t.

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

$$\min_{t,x} \left\{ t : c^T x \leq t, l_c \leq Ax \leq u_c, l_x \leq x \leq u_x \forall (c, A, l_c, u_c, l_x, u_x) \in \mathcal{U} \right\}. \tag{7.4}$$

The optimal solution to (7.4) 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 (7.4) of an uncertain linear optimization problem is not a linear optimization problem since (7.4) has infinitely many linear constraints. There are, however, cases when (7.4) can be rewritten equivalently as a linear programming program; in particular, this is the case for interval uncertainty (7.3). Specifically, in the case of (7.3), the robust counterpart of uncertain linear program is equivalent to the following linear program in variables x, y, t:

$$\begin{array}{lrcccl} \text{minimize} & & & t & & \\ \text{subject to} & & & (c^{\mathrm{n}})^T x + (dc)^T y - t & \leq & 0, \quad (a) \\ & l_c^{\mathrm{n}} + dl_c & \leq & (A^{\mathrm{n}})x - (dA)y, & & (b) \\ & & & (A^{\mathrm{n}})x + (dA)y & \leq & u_c^{\mathrm{n}} - du_c, \quad (c) \\ & 0 & \leq & x + y, & & (d) \\ & 0 & \leq & -x + y, & & (e) \\ & l_x^{\mathrm{n}} + dl_x & \leq & x & \leq & u_x^{\mathrm{n}} - du_x, \quad (f) \end{array} \tag{7.5}$$

The origin of (7.5) is quite transparent: The constraints $d - e$ in (7.5) 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 (7.5) 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 (7.3), so that (b) in (7.5) ensures that $l_c \leq Ax$ for all data from the uncertainty set. Similarly, $(c), (a)$ ans f in (7.5) 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 (7.5), 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 (7.5) 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 (7.5). 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 (7.5). 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 (7.5) gives

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

which after some simplifications leads to

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

and

$$\begin{array}{lrcccl}
\text{minimize} & & & t & & \\
\text{subject to} & & & (c^{\mathrm{n}}+dc)^T x + (dc)^T g - t & \leq & 0, \quad (a) \\
& & & 2(A^{\mathrm{n}}+dA)x + (dA)g + f & \leq & u_c^{\mathrm{n}} - du_c + l_c^{\mathrm{n}} + dl_c, \quad (b) \\
& 0 & \leq & 2x + g, & & (c) \\
& 0 & \leq & f, & & (d) \\
& 0 & \leq & g, & & (e) \\
& l_x^{\mathrm{n}} + dl_x & \leq & x & \leq & u_x^{\mathrm{n}} - du_x. \quad (f)
\end{array} \tag{7.6}$$

Please note that this problem has more variables but much fewer constraints than (7.5). Therefore, (7.6) is likely to be solved faster than (7.5). Note too that (7.6).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 Section 7.1.1, 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 (7.5) of this uncertain LP is the linear program

$$\begin{array}{lcl}
\text{(Drug_RC)}: & & \\
\text{maximize} & & \\
t & & \\
\text{subject to} & & \\
t \leq -100 \cdot \texttt{RawI} - 199.9 \cdot \texttt{RawII} + 5500 \cdot \texttt{DrugI} + 6100 \cdot \texttt{DrugII} & & \\
0.01 \cdot 0.995 \cdot \texttt{RawI} + 0.02 \cdot 0.98 \cdot \texttt{RawII} - 0.500 \cdot \texttt{DrugI} - 0.600 \cdot \texttt{DrugII} & \geq & 0 \\
\texttt{RawI} + \texttt{RawII} & \leq & 1000 \\
90.0 \cdot \texttt{DrugI} + 100.0 \cdot \texttt{DrugII} & \leq & 2000 \\
40.0 \cdot \texttt{DrugI} + 50.0 \cdot \texttt{DrugII} & \leq & 800 \\
100.0 \cdot \texttt{RawI} + 199.90 \cdot \texttt{RawII} + 700 \cdot \texttt{DrugI} + 800 \cdot \texttt{DrugII} & \leq & 100000 \\
\texttt{RawI}, \texttt{RawII}, \texttt{DrugI}, \texttt{DrugII} & \geq & 0
\end{array} \tag{7.7}$$

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

Listing 7.3: Output solving problem (7.7).

```
*** 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

di erent 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 di erent 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 uctuations of the contents are taken into account, RawI turns out to be more pro table than RawII.

7.1.4 Random Uncertainty and Ellipsoidal Robust Counterpart

In some cases, it is natural to assume that the perturbations a ecting di erent 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 o ered by the ellipsoidal model of uncertainty. To motivate this model, let us seseee what happens with a particular linear constraint

$$a^T x \quad b \tag{7.8}$$

at a given candidate solution x in the case when the vector a of coe cients of the constraint is a ected by random perturbations:

$$a = a^n + \quad ; \tag{7.9}$$

where a^n is the vector of nominal coe cients and is a random perturbation vector with zero mean and covariance matrix V_a. In this case the value of the left-hand side of (7.8), evaluated at a given x, becomes a random variable with the expected value $(a^n)^T x$ and the standard deviation $\sqrt{x^T V_a x}$. Now let us act as an engineer who believes that the value of a random variable never exceeds its mean plus 3 times the standard deviation; we do not intend to be that speci c and replace 3 in the above rule by a safety parameter which will be in our control. Believing that the value of a random variable never exceeds its mean plus times the standard deviation, we conclude that a safe version of (7.8) is the inequality

$$(a^n)^T x + \sqrt{x^T V_a x} \quad b: \tag{7.10}$$

The word safe above admits a quantitative interpretation: If x satis es (7.10), one can bound from above the probability of the event that random perturbations (7.9) result in violating the constraint (7.8) evaluated at x. The bound in question depends on what we know about the distribution of , e.g.

- We always have the bound given by the Tschebyshev inequality (x satis es (7.10))

$$\text{Prob} \quad a^T x > b \quad \frac{1}{2}:$$

- When is Gaussian, then the Tschebyshev bound can be improved to (x satis es (7.10))

$$\text{Prob} \quad a^T x > b \quad \frac{1}{\sqrt{2}} \int \quad \exp\{ \quad t^2/2\} dt \quad 0:5 \exp\{ \quad {}^2/2\}: \tag{7.11}$$

- Assume that = D , where is certain n m matrix, and = ($_1$; : : : ; $_m$)^T is a random vector with independent coordinates $_1$; : : : ; $_m$ symmetrically distributed in the segment [1; 1]. Setting V = DD^T (V is a natural upper bound on the covariance matrix of), one has: x satis es (7.10) implies

$$\text{Prob} \quad a^T x > b \quad 0:5 \exp\{ \quad {}^2/2\}: \tag{7.12}$$

Please note that in order to ensure the bounds in (7.11) and (7.12)) to be 10 ^{6}, it su ces to set = 5 :13.

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

$$\begin{array}{lll} \text{minimize} & & [c^n + dc]^T x \\ \text{subject to} & (l_c)_i & [a_i^n + da_i]^T x \quad (u_c)_i \; ; \; i = 1 \; ; : : : ; m; \\ & l_x & x \quad u_x \; ; \end{array} \tag{7.13}$$

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

Consider a linear constraint

$$l \leq \sum_{j=1}^{n} a_j x_j \leq u \tag{7.15}$$

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

$$\begin{array}{rcl} l & \leq & \sum_{j=1}^{n} a_j^{\mathrm{n}} x_j - \sum_{j=1}^{n} \sigma_j y_j, \\ & & \sum_{j=1}^{n} a_j^{\mathrm{n}} x_j + \sum_{j=1}^{n} \sigma_j y_j \quad \leq \quad u, \\ 0 & \leq & x_j + y_j, \\ 0 & \leq & -x_j + y_j, \qquad j = 1, \ldots, n. \end{array} \tag{7.16}$$

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^{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 (7.15) with its ellipsoidal robust counterpart

$$\begin{array}{rcl} l & \leq & \sum_{j=1}^{n} a_j^{\mathrm{n}} x_j - \Omega z, \\ & & \sum_{j=1}^{n} a_j^{\mathrm{n}} x_j + \Omega z \quad \leq \quad u, \\ 0 & \leq & z - \sqrt{\sum_{j=1}^{n} \sigma_j^2 x_j^2}. \end{array} \tag{7.17}$$

Please note that with the model of random perturbations, a vector x satisfying (7.17) satisfies a realization of (7.15) 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 (7.15). On the other hand, the uncertainty set associated with (7.16) 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 (7.17) 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 Ω, say $\Omega = 6$, and $n \geq 40$, the ellipsoid $E(\Omega)$ in its diameter, typical linear sizes, volume, etc. is incomparably less than the box B, the difference becoming more dramatic the larger the dimension n of the box and the ellipsoid. It follows that the ellipsoidal robust counterpart (7.17) of the randomly perturbed uncertain constraint (7.15) is much less conservative than the interval robust counterpart (7.16), 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

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

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 δ_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 σ and is in this sense the most risky. To reduce the risk, one can use the Robust Counterpart approach which results in the following optimization problems.

The Interval Model of Uncertainty:

$$\begin{array}{lrcll} \text{maximize} & & t & & \\ \text{subject to} & 0 & \leq & -t + \sum_{j=1}^{n}(\delta_j - \sigma_j)x_j, & \\ & & & \sum_{j=1}^{n} x_j & = & 1, \\ & 0 & \leq & x_j, & & j = 1, \ldots, n \end{array} \tag{7.18}$$

and

The ellipsoidal Model of Uncertainty:}

$$\begin{array}{lrcll} \text{maximize} & & t & & \\ \text{subject to} & 0 & \leq & -t + \sum_{j=1}^{n}(\delta_j)x_j - \Omega z, & \\ & 0 & \leq & z - \sqrt{\sum_{j=1}^{n} \sigma_j^2 x_j^2}, & \\ & & & \sum_{j=1}^{n} x_j & = & 1, \\ & 0 & \leq & x_j, & & j = 1, \ldots, n. \end{array} \tag{7.19}$$

Note that the problem (7.19) is essentially the risk-averted portfolio model proposed in mid-50's by Markowitz.

The solution of (7.18) 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 (7.19) prescribes a quite reasonable diversification of investments which allows to get much better total return than (7.18) 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:

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

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 7.4 a MATLAB script which builds the associated problem (7.19), 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 7.4: Script that implements problem (7.19).

```
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
```

[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

```
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 7.5: 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
```

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 (7.18) 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 (7.19) 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.

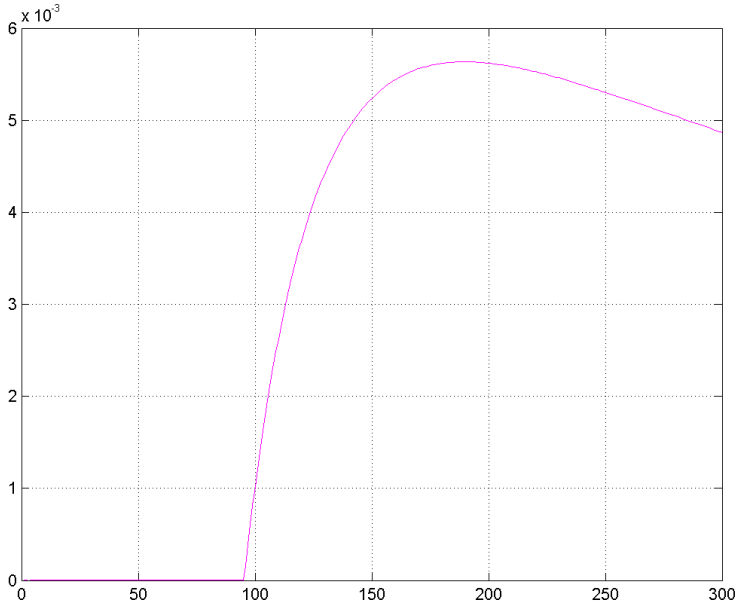

Fig. 7.1: Distribution of investments among the assets in the optimal solution of.

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 (7.15) 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 (7.16);
- 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 (7.17).

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 (7.15) corresponding to the uncertainty set $\mathcal{U}(\Omega)$ is still given by a system of linear and conic quadratic inequalities, specifically the system

$$\begin{array}{rcccl} l & \leq & \sum_{j=1}^n a_j^{\mathrm{n}} x_j - \sum_{j=1}^n \sigma_j y_j - \Omega\sqrt{\sum_{j=1}^n \sigma_j^2 u_j^2}, & & \\ & & \sum_{j=1}^n a_j^{\mathrm{n}} x_j + \sum_{j=1}^n \sigma_j z_j + \Omega\sqrt{\sum_{j=1}^n \sigma_j^2 v_j^2} & \leq & u, \\ -y_j & \leq & x_j - u_j & \leq & y_j, j = 1, \ldots, n, \\ -z_j & \leq & x_j - v_j & \leq & z_j, j = 1, \ldots, n. \end{array} \tag{7.20}$$

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 (7.20), then the probability for x to satisfy a realization of (7.15) 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 (7.20). 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.

7.2 Geometric (posynomial) Optimization

7.2.1 Problem Definition

A geometric optimization problem can be stated as follows

$$\begin{array}{lcl} \text{minimize} & \sum_{k\in J_0} c_k \prod_{j=0}^{n-1} t_j^{a_{kj}} & \\ \text{subject to} & \sum_{k\in J_i} c_k \prod_{j=0}^{n-1} t_j^{a_{kj}} \leq 1, & i = 1, \ldots, m, \\ & t > 0, & \end{array} \tag{7.21}$$

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 (7.21) 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

$$\begin{array}{lll} \text{minimize} & \sum_{k\in J_0} c_k e^{\sum_{j=0}^{n-1} a_{kj}x_j} & \\ \text{subject to} & \sum_{k\in J_i} c_k e^{\sum_{j=0}^{n-1} a_{kj}x_j} & \leq \quad 1, \quad i=1,\ldots,m, \end{array} \tag{7.22}$$

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 (7.22) is non-separable. However, using

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

we obtain the separable problem

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

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 (7.21), 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

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

CHAPTER

NINE

THE OPTIMIZERS FOR CONTINUOUS PROBLEMS

The most essential part of **MOSEK** is the optimizers. Each optimizer is designed to solve a particular class of problems, i.e. linear, conic, or general nonlinear problems. The purpose of the present chapter is to discuss which optimizers are available for the continuous problem classes and how the performance of an optimizer can be tuned, if needed. This chapter deals with the optimizers for continuous problems with no integer variables.

When the optimizer is called, it roughly performs the following steps:

1. Presolve: Preprocessing to reduce the size of the problem.
2. Dualizer: Choosing whether to solve the primal or the dual form of the problem.
3. Scaling: Scaling the problem for better numerical stability.
4. Optimize: Solve the problem using selected method.

The first three preprocessing steps are transparent to the user, but useful to know about for tuning purposes. In general, the purpose of the preprocessing steps is to make the actual optimization more efficient and robust.

Using multiple threads

The interior-point optimizers in **MOSEK** have been parallelized. This means that if you solve linear, quadratic, conic, or general convex optimization problem using the interior-point optimizer, you can take advantage of multiple CPU's.

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

The speed-up obtained when using multiple threads is highly problem and hardware dependent, and consequently, it is advisable to compare single threaded and multi threaded performance for the given problem type to determine the optimal settings.

For small problems, using multiple threads is not be worthwhile and may even be counter productive.

9.1 Presolve

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

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

5. reduce the size of the optimization problem in general.

After the presolved problem has been optimized the solution is automatically postsolved so that the returned solution is valid for the original problem. Hence, the presolve is completely transparent. For further details about the presolve phase, please see [AA95] and [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 then it is suggested 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 presolve off may help.

Since all computations are done in finite prescision then the presolve employs some tolerances when concluding a variable is fixed or constraint is redundant. If it happens that **MOSEK** incorrectly concludes a problem is primal or dual infeasible, then it is worthwhile to try to reduce the parameters *MSK_DPAR_PRESOLVE_TOL_X* and *MSK_DPAR_PRESOLVE_TOL_S*. However, if 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 practise to build models without linear dependencies. If the linear dependencies are removed at the modeling stage, the linear dependency check can safely be disabled by setting the parameter *MSK_IPAR_PRESOLVE_LINDEP_USE* to *MSK_OFF*.

Dualizer

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

- *MSK_IPAR_INTPNT_SOLVE_FORM*: In case of the interior-point optimizer.
- *MSK_IPAR_SIM_SOLVE_FORM*: In case of the simplex optimizer.

Note that currently only linear problems may be dualized.

Scaling

Problems containing data with large and/or small coefficients, say $1.0e+9$ or $1.0e-7$, are often hard to solve. Significant digits may be truncated in calculations with finite precision, which can result in the optimizer relying on inaccurate calculations. Since computers work in finite precision, extreme coefficients should be avoided. In general, data around the same order of magnitude is preferred, and we will refer to a problem, satisfying this loose property, as being well-scaled If the problem is not well scaled, **MOSEK** will try to scale (multiply) constraints and variables by suitable constants. **MOSEK** solves the scaled problem to improve the numerical properties.

The scaling process is transparent, i.e. the solution to the original problem is reported. It is important to be aware that the optimizer terminates when the termination criterion is met on the scaled problem, therefore significant primal or dual infeasibilities may occur after unscaling for badly scaled problems. The best solution of this issue is to reformulate the problem, making it better scaled.

By default **MOSEK** heuristically chooses a suitable scaling. The scaling for interior-point and simplex optimizers can be controlled with the parameters *MSK_IPAR_INTPNT_SCALING* and *MSK_IPAR_SIM_SCALING* respectively.

9.2 Linear Optimization

9.2.1 Optimizer Selection

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

9.2.2 The Interior-point Optimizer

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

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

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

Interior-point Termination Criterion

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

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

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

to homogeneous model is generated where

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

Whenever the trial solution satisfies the criterion

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

the interior-point optimizer is terminated and

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

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

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

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 ε_i controls the quality of the approximation. A smaller value means a better approximation.

Finally, if

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

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

It is possible to adjust the tolerances ε_p, ε_d, ε_g and ε_i using parameters; see Table 9.1 for details.

Table 9.1: Parameters employed in termination criterion

ToleranceParameter	name
ε_p	*MSK_DPAR_INTPNT_TOL_PFEAS*
ε_d	*MSK_DPAR_INTPNT_TOL_DFEAS*
ε_g	*MSK_DPAR_INTPNT_TOL_REL_GAP*
ε_i	*MSK_DPAR_INTPNT_TOL_INFEAS*

The default values of the termination tolerances are chosen such that for a majority of problems appearing in practice it is not possible to achieve much better accuracy. Therefore, tightening the tolerances usually is not worthwhile. However, an inspection of (9.5) reveals that quality of the solution is dependent on $\|b\|_\infty$ and $\|c\|_\infty$; the smaller the norms are, the better the solution accuracy.

The interior-point method as implemented by **MOSEK** will converge toward optimality and primal and dual feasibility at the same rate [And09]. This means that if the optimizer is stopped prematurely then it is very unlikely that either the primal or dual solution is feasible. Another consequence is that in most cases all the tolerances, ε_p, ε_d and ε_g, 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 (9.5). A solution is defined as near optimal if scaling ε_p, ε_d and ε_g by any number $\varepsilon_n \in [1.0, +\infty]$ conditions (9.5) are satisfied.

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

The basis identification discussed in Section 9.2.2.2 requires an optimal solution to work well; hence basis identification should be turned off if the termination criterion is relaxed.

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

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 relaxation 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:

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

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 phase be short. However, in some cases for nasty problems e.g. 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 can be tuned by the *MSK_IPAR_BI_CLEAN_OPTIMIZER* parameter i.e. primal or dual simplex, and the maximum number of iterations can be set by the *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 presented:

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

The first line displays the number of threads used by the optimizer and second line tells that the optimizer 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 Section 9.2.2 the columns of the iteration log have the following meaning:

- `ITE`: Iteration index.
- `PFEAS`: $\left\|Ax^k - b\tau^k\right\|_\infty$. The numbers in this column should converge monotonically towards to zero but may stall at low level due to rounding errors.
- `DFEAS`: $\left\|A^T y^k + s^k - c\tau^k\right\|_\infty$. The numbers in this column should converge monotonically toward to zero but may stall at low level due to rounding errors.

- GFEAS: $|-c^T x^k + b^T y^k - \kappa^k|$. The numbers in this column should converge monotonically toward to 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 monotonically to zero.
- TIME: Time spend since the optimization started.

9.2.3 The simplex Based Optimizer

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

The simplex optimizer uses a different method that allows exploiting an initial guess for the optimal solution to reduce the solution time. Depending on the problem it may be faster or slower to use an initial guess; see section 9.2.4 for a discussion.

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

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 Section 14.1 and 14.1.1 for a definition of the primal and dual problem. Due to the fact that computations are performed in finite precision **MOSEK** allows violation of primal and dual feasibility within certain tolerances. The user can control the allowed primal and dual tolerances with the parameters *MSK_DPAR_BASIS_TOL_X* and *MSK_DPAR_BASIS_TOL_S*.

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.

Setting the parameter *MSK_IPAR_OPTIMIZER* to *MSK_OPTIMIZER_FREE_SIMPLEX* instructs **MOSEK** to select automatically between the primal and the dual simplex optimizers. Hence, **MOSEK** tries to choose the best optimizer for the given problem and the available solution.

By default **MOSEK** uses presolve when performing a 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** counts a "numerical unexpected behavior" event inside the optimizer as a set-back The user can define how many set-backs the optimizer accepts; if that number is exceeded, the optimization will be aborted. Set-backs are implemented to avoid long sequences where the optimizer tries to recover from an unstable situation.

Set-backs are, for example, repeated singularities when factorizing the basis matrix, repeated loss of feasibility, degeneracy problems (no progress in objective) and other events indicating numerical difficulties. If the simplex optimizer encounters a lot of set-backs the problem is usually badly scaled; in such

a situation try to reformulate into a better scaled problem. Then, if a lot of set-backs still occur, trying one or more of the following suggestions may be worthwhile:

- Raise tolerances for allowed primal or dual feasibility: Hence, increase the value of
  - *MSK_DPAR_BASIS_TOL_X*, and
  - *MSK_DPAR_BASIS_TOL_S*.
- Raise or lower pivot tolerance: Change the *MSK_DPAR_SIMPLEX_ABS_TOL_PIV* parameter.
- Switch optimizer: Try another optimizer.
- Switch off crash: Set both *MSK_IPAR_SIM_PRIMAL_CRASH* and *MSK_IPAR_SIM_DUAL_CRASH* to 0.
- Experiment with other pricing strategies: Try different values for the parameters
  - *MSK_IPAR_SIM_PRIMAL_SELECTION* and
  - *MSK_IPAR_SIM_DUAL_SELECTION*.
- If you are using warm-starts, in rare cases switching off this feature may improve stability. This is controlled by the *MSK_IPAR_SIM_HOTSTART* parameter.
- Increase maximum set backs allowed controlled by *MSK_IPAR_SIM_MAX_NUM_SETBACKS*.
- If the problem repeatedly becomes infeasible try switching off the special degeneracy handling. See the parameter *MSK_IPAR_SIM_DEGEN* for details.

9.2.4 The Interior-point or the Simplex Optimizer?

Given a linear optimization problem, which optimizer is the best: The primal simplex, the dual simplex or the interior-point optimizer?

It is impossible to provide a general answer to this question. However, the interior-point optimizer behaves more predictably: it tends to use between 20 and 100 iterations, almost independently of problem size, but cannot perform warm-start, while simplex can take advantage of an initial solution, but is less predictable for cold-start. The interior-point optimizer is used by default.

9.2.5 The Primal or the Dual Simplex Variant?

MOSEK provides both a primal and a dual simplex optimizer. Predicting which simplex optimizer is faster is impossible, however, in recent years the dual optimizer has seen several algorithmic and computational improvements, which, in our experience, makes it faster on average than the primal simplex optimizer. Still, it depends much on the problem structure and size.

Setting the *MSK_IPAR_OPTIMIZER* parameter to *MSK_OPTIMIZER_FREE_SIMPLEX* instructs **MOSEK** to choose which simplex optimizer to use automatically.

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

9.3 Conic Optimization

9.3.1 The Interior-point Optimizer

For conic optimization problems only an interior-point type optimizer is available. The interior-point optimizer is an implementation of the so-called homogeneous and self-dual algorithm. For a detailed description of the algorithm, please see [ART03] .

Interior-point Termination Criteria

The parameters controlling when the conic interior-point optimizer terminates are shown in Table 9.2.

Table 9.2: Parameters employed in termination criterion.

Parameter name	Purpose
MSK_DPAR_INTPNT_CO_TOL_PFEAS	Controls primal feasibility
MSK_DPAR_INTPNT_CO_TOL_DFEAS	Controls dual feasibility
MSK_DPAR_INTPNT_CO_TOL_REL_GAP	Controls relative gap
MSK_DPAR_INTPNT_TOL_INFEAS	Controls when the problem is declared infeasible
MSK_DPAR_INTPNT_CO_TOL_MU_RED	Controls when the complementarity is reduced enough

9.4 Nonlinear Convex Optimization

9.4.1 The Interior-point Optimizer

For quadratic, quadratically constrained, and general convex optimization problems an interior-point type optimizer is available. The interior-point optimizer is an implementation of the homogeneous and self-dual algorithm. For a detailed description of the algorithm, please see [AY98] , [AY99] .

The Convexity Requirement

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

The following parameters are available to control the convexity check:

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

The Differentiabilty Requirement

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

For instance, the function

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

is differentiable everywhere whereas the function

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

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

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

$$x \geq 0$$

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

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

$$f(x) = \frac{1}{x} + \frac{1}{1-x}. \tag{9.6}$$

In this case the bounds

$$0 \leq x \leq 1$$

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

For efficiency reasons it may be better to reformulate the problem than to force **MOSEK** to observe ranged bounds strictly. For instance, (9.6) can be reformulated as follows

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

Interior-point Termination Criteria

The parameters controlling when the general convex interior-point optimizer terminates are shown in Table 9.3.

Table 9.3: 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

9.5 Using Multiple Threads in an Optimizer

If multiple cores are available then it is possible for **MOSEK** to take advantage of them to speed up the computation. However, please note the speedup achieved is going to be dependent on the problem characteristics e.g. the size of problem. Typically for smallish problems no speedup is obtained by exploiting multiple cores. In fact forcing **MOSEK** to use one core can increase speed because parallel overhead is avoided.

9.5.1 Thread Safety

The **MOSEK** API is thread-safe provided that a task is only modified or accessed from one thread at any given time. Also accessing two or more separate tasks from threads at the same time is safe. Sharing an environment between threads is safe.

9.5.2 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 qualification 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.

9.5.3 The Parallelized Interior-point Optimizer

By default the interior-point optimizer exploits multiple cores using multithreading. Hence, big tasks such as large dense matrix multiplication may be divided into several independent smaller tasks that can be computed independently. However, there is a computational overhead associated with exploiting multiple threads e.g. cost of the additional coordination etc. Therefore, it may be advantageous to turn off the mutithreading for smallish problem using the parameter *MSK_IPAR_INTPNT_MULTI_THREAD*.

Moreover, when the interior-point optimizer is allowed to exploit multiple threads, then the parameter *MSK_IPAR_NUM_THREADS* controls the maximum number of threads (and therefore the number of cores) that **MOSEK** will employ.

CHAPTER

TEN

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. MOSEK can solve mixed-integer

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

problems.

Readers unfamiliar with integer optimization are recommended to consult some relevant literature, e.g. the book [Wol98] by Wolsey.

10.1 Some Concepts and Facts Related to Mixed-integer Optimization

It is important to understand that in a worst-case scenario, the time required to solve integer optimization problems grows exponentially with the size of the problem. For instance, assume that a problem contains n binary variables, then the time required to solve the problem in the worst case may be proportional to 2^n. The value of 2^n is huge even for moderate values of n.

In practice this implies that the focus should be on computing a near optimal solution quickly rather than 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.

The mixed-integer optimization problem

$$\begin{array}{rcl} z & = & \text{minimize} \quad c^T x \\ & & \text{subject to} \quad Ax = b; \\ & & \qquad x \geq 0 \\ & & \qquad x_j \in \mathbb{Z}; \quad \forall j \in J; \end{array} \tag{10.1}$$

has the continuous relaxation

$$\begin{array}{rcl} \underline{z} & = & \text{minimize} \quad c^T x \\ & & \text{subject to} \quad Ax = b; \\ & & \qquad x \geq 0 \end{array} \tag{10.2}$$

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

There are two important observations about the continuous relaxation. First, the continuous relaxation is usually much faster to optimize than the mixed-integer problem. Secondly if $\hat{x}$ is any feasible solution to (10.1) and

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

is satisfied. If this is the case, the integer optimizer terminates and reports the integer feasible solution as an optimal solution. Please note that $\underline{z}$ is a valid lower bound determined by the integer optimizer during the solution process, i.e.

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

The lower bound $\underline{z}$ normally increases during the solution process.

10.3.1 Relaxed Termination

If an optimal solution cannot be located within a reasonable time, it may be advantageous to employ a relaxed termination criterion after some time. Whenever the integer optimizer locates an integer feasible solution and has spent at least the number of seconds defined by the *MSK_DPAR_MIO_DISABLE_TERM_TIME* parameter on solving the problem, it will check whether the criterion

$$\bar{z} - \underline{z} \leq \max(\delta_4, \delta_5 \max(10^{-10}, |\bar{z}|))$$

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

10.4 Parameters Affecting the Termination of the Integer Optimizer.

All δ tolerances can be adjusted using suitable parameters — see Table 10.1.

Table 10.1: Tolerances for the mixed-integer optimizer.

Tolerance	Parameter name
δ_1	*MSK_DPAR_MIO_TOL_ABS_RELAX_INT*
δ_2	*MSK_DPAR_MIO_TOL_ABS_GAP*
δ_3	*MSK_DPAR_MIO_TOL_REL_GAP*
δ_4	*MSK_DPAR_MIO_NEAR_TOL_ABS_GAP*
δ_5	*MSK_DPAR_MIO_NEAR_TOL_REL_GAP*

In Table 10.2 some other parameters affecting the integer optimizer termination criterion are shown. Please note that if the effect of a parameter is delayed, the associated termination criterion is applied only after some time, specified by the *MSK_DPAR_MIO_DISABLE_TERM_TIME* parameter.

Table 10.2: 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.

10.5 How to Speed Up the Solution Process

As mentioned previously, in many cases it is not possible to find an optimal solution to an integer optimization problem in a reasonable amount of time. Some suggestions to reduce the solution time are:

- Relax the termination criterion: In case the run time is not acceptable, the first thing to do is to relax the termination criterion — see Section 10.3 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].

10.6 Understanding Solution Quality

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

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

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

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

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

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

CHAPTER

ELEVEN

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.

11.1 General Characteristics

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

Constraints

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

Bounds

- `upper bd` The number of upper bounded variables, $x_j \leq u_j^x$
- `lower bd` The number of lower bounded variables, $l_k^x \leq x_j$
- `ranged` The number of ranged variables, $l_k^x \leq x_j \leq u_j^x$
- `fixed` The number of fixed variables, $l_k^x = x_j = u_j^x$

- `free` The number of free variables

`Variables`

- `cont` The number of continuous variables, $x_j \in \mathbb{R}$
- `bin` The number of binary variables, $x_j \in \{0, 1\}$
- `int` The number of general integer variables, $x_j \in \mathbb{Z}$

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

11.2 Objective

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

The range is displayed using three terms:

- `min |c|` The minimum absolute value among all coeffecients
- `min |c|>0` The minimum absolute value among the nonzero coefficients
- `max |c|` The maximum absolute value among the coefficients

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

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

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

11.3 Linear Constraints

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

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

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

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

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

11.4 Constraint and Variable Bounds

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

11.5 Quadratic Constraints

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

Note: Quadratic constraints may also have a linear part, but that will be included in the linear constraints survey; this means that if a problem has one or more pure quadratic constraints, part three of the survey will report anleq al 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.

11.6 Conic Constraints

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

CHAPTER

TWELVE

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.

12.1 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. 12.1.

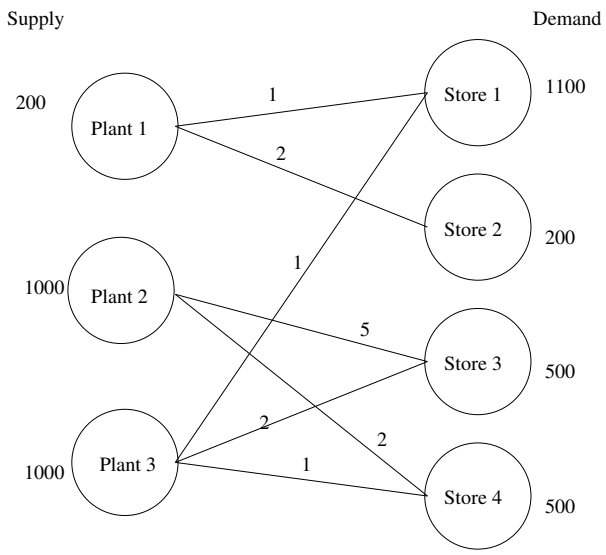

Fig. 12.1: Supply, demand and cost of transportation.

The problem represented in Fig. 12.1 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:

$$\begin{array}{lccccccccccccccl}
\text{minimize} & x_{11} & + & 2x_{12} & + & 5x_{23} & + & 2x_{24} & + & x_{31} & + & 2x_{33} & + & x_{34} & & \\
\text{subject to} & x_{11} & + & x_{12} & & & & & & & & & & & \leq & 200, \\
& & & & & x_{23} & + & x_{24} & & & & & & & \leq & 1000, \\
& & & & & & & & & x_{31} & + & x_{33} & + & x_{34} & \leq & 1000, \\
& x_{11} & & & & & & & + & x_{31} & & & & & = & 1100, \\
& & & x_{12} & & & & & & & & & & & = & 200, \\
& & & & & x_{23} & + & & & & & x_{33} & & & = & 500, \\
& & & & & & & x_{24} & + & & & & & x_{34} & = & 500, \\
& x_{ij} \geq 0. & & & & & & & & & & & & & &
\end{array} \tag{12.1}$$

Solving problem (12.1) 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 certi cate of the infeasibility was found. The certificate is returned in place of the solution to the problem.

12.2 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 (Section 12.4) 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.

12.3 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.

12.3.1 A cautionary note

The problem

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

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.

12.4 The Infeasibility Report

MOSEK includes functionality for diagnosing the cause of a primal or a dual infeasibility. It can be turned on by setting the *MSK_IPAR_INFEAS_REPORT_AUTO* to *MSK_ON*. This causes **MOSEK** to print a report on variables and constraints involved in the infeasibility.

The *MSK_IPAR_INFEAS_REPORT_LEVEL* parameter controls the amount of information presented in the infeasibility report. The default value is 1.

12.4.1 Example: Primal Infeasibility

We will keep working with the problem (12.1) written in LP format:

Listing 12.1: The code for problem (12.1).

```
\
\ 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
```

12.4.2 Example: Dual Infeasibility

The following problem is dual to (12.1) and therefore it is dual infeasible.

Listing 12.2: The dual of problem (12.1).

```
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 Section 14.1.2.2) 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
```

CHAPTER

THIRTEEN

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.

13.1 Sensitivity Analysis for Linear Problems

13.1.1 The Optimal Objective Value Function

Assume that we are given the problem

$$\begin{array}{rcllllll} z(l^c, u^c, l^x, u^x, c) & = & \text{minimize} & & & c^T x & & \\ & & \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & & & l^x & \leq & x & \leq & u^x, \end{array} \tag{13.1}$$

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

$$f_{l_i^c}(\beta) \tag{13.2}$$

shows the optimal objective value as a function of β. Please note that a change in β corresponds to a perturbation in l_i^c and hence (13.2) 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 (13.2) is a piecewise linear and convex function, i.e. its graph may look like in Fig. 13.1 and Fig. 13.2.

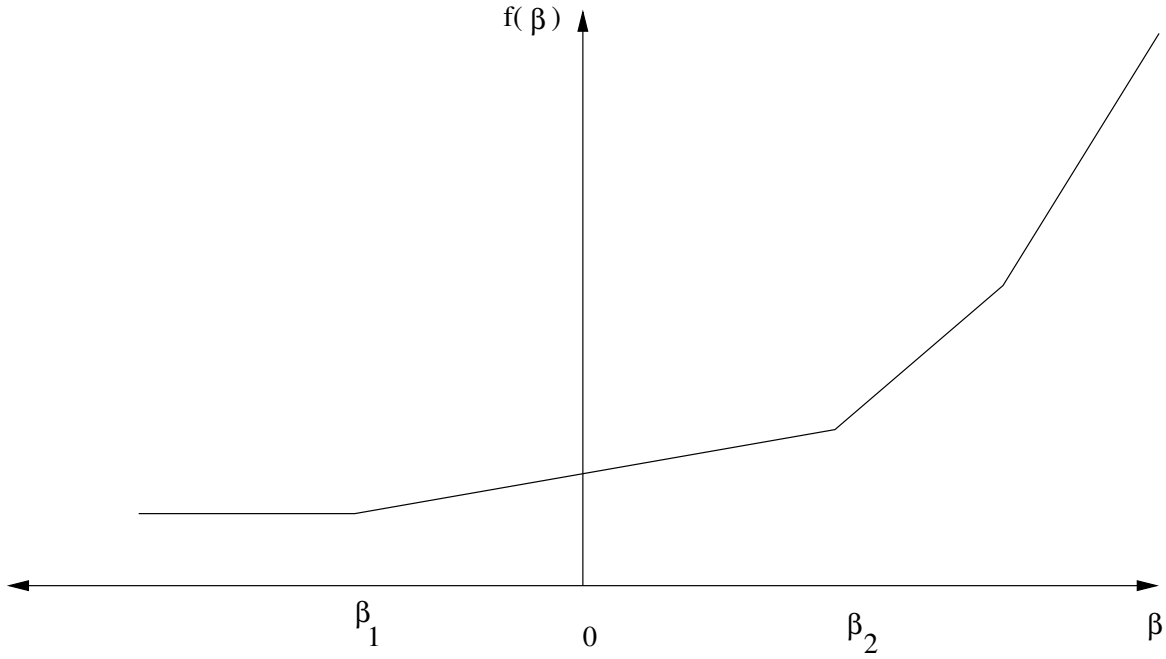

Fig. 13.1: $\beta = 0$ is in the interior of linearity interval.

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

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

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

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

for which

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

Since $f_{l_i^c}$ is not a smooth function $f'_{l_i^c}$ may not be defined at 0, as illustrated in Fig. 13.2. 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 (13.1) 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.

13.1.3 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 (13.1), i.e. x^* and $((s_l^c)^*, (s_u^c)^*, (s_l^x)^*, (s_u^x)^*)$ the optimal objective value is given by

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

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

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

and

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

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

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

and

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

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

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

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

and

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

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 $[\ _1;\ _2]$ for a c_j is computed as follows:

$$
\begin{array}{lll}
_1 = & \text{minimize} & \\
 & \text{subject to} & A^T(s_l^c \quad s_u^c) + s_l^x \quad s_u^x = c + e_j; \\
 & & (l^c)^T(s_l^c) \quad (u^c)^T(s_u^c) + (l^x)^T(s_l^x) \quad (u^x)^T(s_u^x) \quad _1 \quad z; \\
 & & s_l^c; s_u^c; s_l^c; s_u^x \quad 0
\end{array}
$$

and

$$
\begin{array}{lll}
_2 = & \text{maximize} & \\
 & \text{subject to} & A^T(s_l^c \quad s_u^c) + s_l^x \quad s_u^x = c + e_j; \\
 & & (l^c)^T(s_l^c) \quad (u^c)^T(s_u^c) + (l^x)^T(s_l^x) \quad (u^x)^T(s_u^x) \quad _2 \quad z; \\
 & & s_l^c; s_u^c; s_l^c; s_u^x \quad 0:
\end{array}
$$

13.1.4 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. 13.3.

Fig. 13.3: 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

$$
\begin{array}{lllllllll}
x_{11} + x_{12} & & & & & & & & 400; \\
 & & x_{23} + x_{24} & & & & & & 1200; \\
 & & & & x_{31} + x_{33} + x_{34} & & & & 1000; \\
x_{11} & & & & + x_{31} & & & = & 800; \\
 & x_{12} & & & & & & = & 100; \\
 & & x_{23} + & & & x_{33} & & = & 500; \\
 & & & x_{24} + & & & x_{34} & = & 500; \\
x_{11}; & x_{12}; & x_{23}; & x_{24}; & x_{31}; & x_{33}; & x_{34} & & 0:
\end{array}
\tag{13.3}
$$

The sensitivity parameters are shown in Table 13.1 and Table 13.2 for the basis type analysis and in Table 13.3 and Table 13.4 for the optimal partition type analysis.

Table 13.1: Ranges and shadow prices related to bounds on constraints and variables: results for the basis type sensitivity analysis.

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

Table 13.2: Ranges and shadow prices related to bounds on constraints and variables: results for the optimal partition type sensitivity analysis.

Con.	β_1	β_2	σ_1	σ_2
1	-300.00	500.00	3.00	1.00
2	-700.00	$+\infty$	-0.00	-0.00
3	-500.00	500.00	3.00	1.00
4	-500.00	500.00	2.00	4.00
5	-100.00	300.00	3.00	5.00
6	-500.00	700.00	3.00	5.00
7	-500.00	700.00	2.00	2.00
Var.	β_1	β_2	σ_1	σ_2
x_{11}	$-\infty$	300.00	0.00	0.00
x_{12}	$-\infty$	100.00	0.00	0.00
x_{23}	$-\infty$	500.00	0.00	2.00
x_{24}	$-\infty$	500.00	0.00	0.00
x_{31}	$-\infty$	500.00	0.00	0.00
x_{33}	$-\infty$	500.00	0.00	0.00
x_{34}	$-\infty$	500.00	0.00	2.00

Table 13.3: Ranges and shadow prices related to the objective coefficients: results for the basis type sensitivity analysis.

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

Table 13.4: Ranges and shadow prices related to the objective coefficients: results for the optimal partition type sensitivity analysis.

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

Examining the results from the 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

$$\sigma_1 \beta = 3\beta.$$

13.2 Sensitivity Analysis with MOSEK

The following describe sensitivity analysis from the MATLAB toolbox.

13.2.1 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`.

13.2.2 *prisen*

The field `prisen` is structured as follows:

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

13.2.3 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 (13.3). 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 13.1: A script to perform sensitivity analysis on problem (13.3).

```
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 13.1 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

FOURTEEN

PROBLEM FORMULATION AND SOLUTIONS

In this chapter we will discuss the following issues:

- The formal definitions of the problem types that **MOSEK** can solve.
- The solution information produced by **MOSEK**.
- The information produced by **MOSEK** if the problem is infeasible.

14.1 Linear Optimization

A linear optimization problem can be written as

$$\begin{array}{lcccl} \text{minimize} & & c^T x + c^f & & \\ \text{subject to} & l^c & \leq \quad Ax & \leq & u^c, \\ & l^x & \leq \quad x & \leq & u^x, \end{array} \tag{14.1}$$

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 (14.1). If (14.1) has at least one primal feasible solution, then (14.1) is said to be (primal) feasible.

In case (14.1) does not have a feasible solution, the problem is said to be (primal) infeasible

14.1.1 Duality for Linear Optimization

Corresponding to the primal problem (14.1), there is a dual problem

$$\begin{array}{lcccl} \text{maximize} & & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f & & \\ & & A^T y + s_l^x - s_u^x & = & c, \\ \text{subject to} & & -y + s_l^c - s_u^c & = & 0, \\ & & s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0. \end{array} \tag{14.2}$$

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 (14.2). If (14.2) has at least one feasible solution, then (14.2) 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 (14.1) and $(y, s_l^c, s_u^c, s_l^x, s_u^x)$ is a solution to the corresponding dual problem (14.2).

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,

$$\begin{array}{l} c^T x^* + c^f - \left\{ (l^c)^T (s_l^c)^* - (u^c)^T (s_u^c)^* + (l^x)^T (s_l^x)^* - (u^x)^T (s_u^x)^* + c^f \right\} \\ \quad = \sum_{i=0}^{m-1} \left[(s_l^c)_i^* ((x_i^c)^* - l_i^c) + (s_u^c)_i^* (u_i^c - (x_i^c)^*) \right] \\ \quad + \sum_{j=0}^{n-1} \left[(s_l^x)_j^* (x_j - l_j^x) + (s_u^x)_j^* (u_j^x - x_j^*) \right] \geq 0 \end{array} \tag{14.3}$$

where the first relation can be obtained by transposing and multiplying the dual constraints (14.2) 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 (14.1) 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.

14.1.2 Infeasibility for Linear Optimization

Primal Infeasible Problems

If the problem (14.1) 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

$$\begin{array}{ll} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x \\ \text{subject to} & \\ & A^T y + s_l^x - s_u^x = 0, \\ & y + s_l^c - s_u^c = 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0, \end{array} \tag{14.4}$$

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 (14.4) 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 (14.4) is unbounded, and that its dual is infeasible. As the constraints to the dual of (14.4) are identical to the constraints of problem (14.1), we thus have that problem (14.1) is also infeasible.

Dual Infeasible Problems

If the problem (14.2) 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

$$\begin{array}{lll} \text{minimize} & & c^T x \\ \text{subject to} & \hat{l}^c \leq & Ax \leq \hat{u}^c, \\ & \hat{l}^x \leq & x \leq \hat{u}^x, \end{array} \tag{14.5}$$

where

$$\hat{l}_i^c = \begin{cases} 0 & \text{if } l_i^c > -1, \\ -1 & \text{otherwise,} \end{cases} \quad \text{and} \quad \hat{u}_i^c := \begin{cases} 0 & \text{if } u_i^c < 1, \\ 1 & \text{otherwise,} \end{cases}$$

and

$$\hat{l}_j^x = \begin{cases} 0 & \text{if } l_j^x > -1, \\ -1 & \text{otherwise,} \end{cases} \quad \text{and} \quad \hat{u}_j^x := \begin{cases} 0 & \text{if } u_j^x < 1, \\ 1 & \text{otherwise,} \end{cases}$$

such that

$$c^T x < 0.$$

Such a solution implies that (14.5) is unbounded, and that its dual is infeasible. As the constraints to the dual of (14.5) are identical to the constraints of problem (14.2), we thus have that problem (14.2) is also infeasible.

Primal and Dual Infeasible Case

In case that both the primal problem (14.1) and the dual problem (14.2) 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 (14.1) is maximization, i.e.

$$\begin{array}{lll} \text{maximize} & & c^T x + c^f \\ \text{subject to} & l^c & Ax \quad u^c; \\ & l^x & x \quad u^x; \end{array}$$

the objective sense of the dual problem changes to minimization, and the domain of all dual variables changes sign in comparison to (14.2). The dual problem thus takes the form

$$\begin{array}{ll} \text{minimize} & (l^c)^T s_l^c \quad (u^c)^T s_u^c + (l^x)^T s_l^x \quad (u^x)^T s_u^x + c^f \\ \text{subject to} & \\ & A^T y + s_l^x \quad s_u^x = c; \\ & y + s_l^c \quad s_u^c = 0; \\ & s_l^c; s_u^c; s_l^x; s_u^x \quad 0: \end{array}$$

This means that the duality gap, de ned in (14.3) 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 certi cate will be reported by MOSEK as a solution to the system

$$\begin{array}{l} A^T y + s_l^x \quad s_u^x = 0; \\ y + s_l^c \quad s_u^c = 0; \\ s_l^c; s_u^c; s_l^x; s_u^x \quad 0; \end{array} \tag{14.6}$$

such that the objective value is strictly negative

$$(l^c)^T (s_l^c) \quad (u^c)^T (s_u^c) + (l^x)^T (s_l^x) \quad (u^x)^T (s_u^x) < 0:$$

Similarly, the certi cate of dual infeasibility is an x satisfying the requirements of (14.5) such that $c^T x > 0$.

14.2 Conic Quadratic Optimization

Conic quadratic optimization is an extension of linear optimization (see Section 14.1) allowing conic domains to be speci ed for subsets of the problem variables. A conic quadratic optimization problem can be written as

$$\begin{array}{lll} \text{minimize} & & c^T x + c^f \\ \text{subject to} & l^c & Ax \quad u^c; \\ & l^x & x \quad u^x; \\ & & x \ 2 \ K; \end{array} \tag{14.7}$$

where set K is a Cartesian product of convex cones, namely $K = K_1 \quad K_p$. Having the domain restriction, $x \ 2 \ K$, is thus equivalent to

$$x^t \ 2 \ K_t \quad R^{n_t};$$

where $x = (x^1; : : : ; x^p)$ is a partition of the problem variables. Please note that the n-dimensional Euclidean space R^n is a cone itself, so simple linear variables are still allowed.

MOSEK supports only a limited number of cones, speci cally:

- The R^n set.
- The quadratic cone:

$$Q^n = \left\{ x \ 2 \ R^n : x_1 \quad \sqrt{\sum_{j=2}^{n} x_j^2}; \right\}$$

- 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].

14.2.1 Duality for Conic Quadratic Optimization

The dual problem corresponding to the conic quadratic optimization problem (14.7) is given by

$$\begin{array}{ll} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f \\ \text{subject to} & \\ & \begin{array}{c} A^T y + s_l^x - s_u^x + s_n^x = c \\ -y + s_l^c - s_u^c = 0, \\ s_l^c, s_u^c, s_l^x, s_u^x \geq 0, \\ s_n^x \in \mathcal{K}^*, \end{array} \end{array} \tag{14.8}$$

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.

14.2.2 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 Section 14.1.2).

Primal Infeasible Problems

If the problem (14.7) 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

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

such that the objective value is strictly positive.

Dual infeasible problems

If the problem (14.8) 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

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

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.

14.3 Semidefinite Optimization

Semide nite optimization is an extension of conic quadratic optimization (see Section 14.2) allowing positive semidefinite matrix variables to be used in addition to the usual scalar variables. A semidefinite optimization problem can be written as

$$\begin{array}{lcccccl} \text{minimize} & & \sum_{j=0}^{n-1} c_j x_j + \sum_{j=0}^{p-1} \langle \overline{C}_j, \overline{X}_j \rangle + c^f & & & & \\ \text{subject to} & l_i^c & \leq & \sum_{j=0}^{n-1} a_{ij} x_j + \sum_{j=0}^{p-1} \langle \overline{A}_{ij}, \overline{X}_j \rangle & \leq & u_i^c, & i = 0, \ldots, m-1 \\ & l_j^x & \leq & x_j & \leq & u_j^x, & j = 0, \ldots, n-1 \\ & & & x \in \mathcal{K}, \overline{X}_j \in \mathcal{S}_+^{r_j}, & & & j = 0, \ldots, p-1 \end{array} \tag{14.9}$$

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].

14.3.1 Duality for Semidefinite Optimization

The dual problem corresponding to the semidefinite optimization problem (14.9) is given by

$$\begin{array}{lll} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f & \\ \text{subject to} & & \\ & c - A^T y + s_u^x - s_l^x = s_n^x, & \\ & \overline{C}_j - \sum_{i=0}^m y_i \overline{A}_{ij} = \overline{S}_j, & j = 0, \ldots, p-1 \\ & s_l^c - s_u^c = y, & \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0, & \\ & s_n^x \in \mathcal{K}^*, \quad \overline{S}_j \in \mathcal{S}_+^{r_j}, & j = 0, \ldots, p-1 \end{array} \tag{14.10}$$

where $A \in \mathbb{R}^{m\times n}$, $A_{ij} = a_{ij}$, which is similar to the dual problem for conic quadratic optimization (see Section 14.2.1), 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.

14.3.2 Infeasibility for Semidefinite Optimization

In case **MOSEK** finds a problem to be infeasible it reports a certificate of the infeasibility. This works exactly as for linear problems (see Section 14.1.2).

Primal Infeasible Problems

If the problem (14.9) 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

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

such that the objective value is strictly positive.

Dual Infeasible Problems

If the problem (14.10) 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

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

where

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

and

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

such that the objective value is strictly negative.

14.4 Quadratic and Quadratically Constrained Optimization

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

$$\begin{array}{lrcccl} \text{minimize} & & & \frac{1}{2}x^T Q^o x + c^T x + c^f & & \\ \text{subject to} & l_k^c & \leq & \frac{1}{2}x^T Q^k x + \sum_{j=0}^{n-1} a_{kj} x_j & \leq & u_k^c, \quad k = 0, \ldots, m-1, \\ & l_j^x & \leq & x_j & \leq & u_j^x, \quad j = 0, \ldots, n-1, \end{array} \tag{14.11}$$

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 fullfiled.

14.4.1 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 formuation if deemed worthwhile potentially leading to (huge) computational savings.

However, instead of relying on the automatic reformulation we recommend to formulate the problem as 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 neeeded for conic problems.
- usually the modeller can do a better reformulation than the automatic method because the modeller can exploit the knowledge of what is being modelled.

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

14.4.2 Duality for Quadratic and Quadratically Constrained Optimization

The dual problem corresponding to the quadratic and quadratically constrained optimization problem (14.11) is given by

$$\begin{array}{l} \text{maximize} \quad (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + \frac{1}{2}x^T \left\{ \sum_{k=0}^{m-1} y_k Q^k - Q^o \right\} x + c^f \\ \text{subject to} \\ \qquad A^T y + s_l^x - s_u^x + \left\{ \sum_{k=0}^{m-1} y_k Q^k - Q^o \right\} x = c, \\ \qquad -y + s_l^c - s_u^c = 0, \\ \qquad s_l^c, s_u^c, s_l^x, s_u^x \geq 0. \end{array} \tag{14.12}$$

The dual problem is related to the dual problem for linear optimization (see Section 14.1.1), 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 (14.11) and the dual problem (14.12).

14.4.3 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 Section 14.1.2).

Primal Infeasible Problems

If the problem (14.11) 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 Section 14.1.2.1).

Dual Infeasible Problems

If the problem (14.12) with all $Q^k = 0$ is infeasible, **MOSEK** will report a certificate of dual infeasibility. The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

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

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.

14.5 General Convex Optimization

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

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

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.
- $f : \mathbb{R}^n \to \mathbb{R}$ is a nonlinear function.
- $g : \mathbb{R}^n \to \mathbb{R}^m$ 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 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

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

14.5.1 Duality for General convex Optimization

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

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

and the dual problem is given by

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

which is equivalent to

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

In this context we use the following definition for scalar functions

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

and accordingly for vector functions

$$\nabla g(x) \quad = \quad \begin{bmatrix} \nabla g_1(x) \\ \vdots \\ \nabla g_m(x) \end{bmatrix}.$$

CHAPTER

FIFTEEN

TOOLBOX REFERENCE

15.1 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.

15.1.1 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 Section 12 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`.
- `version`: Return the **MOSEK** version numbers in `res.version`.

`Parameters`

- `[in]` `cmd` (`string`) – The commands to be executed. By default it takes the value minimize.
- `[in]` `prob` (*prob*) – [optional] a structure containing the problem data.
- `[in]` `param` (`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.
- `[in]` `callback` (*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 Section 15.5.
- `res` (*res*) – [optional] Solution obtained by the interior-point algorithm.

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

Solves the linear least squares problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|Cx-d\|_2^2 \\ \text{subject to} & Ax \leq b, \\ & Bx = c, \\ & l \leq x \leq u. \end{array} \tag{15.2}$$

Parameters

- [in] `C` (`double[][]`) – A matrix. See problem (15.2) for the purpose of the argument.
- [in] `d` (`double[]`) – A vector. See problem (15.2) for the purpose of the argument.
- [in] `A` (`double[][]`) – Constraint matrix for the inequalities. Use $A = []$ if there are no inequalities.
- [in] `b` (`double[]`) – Right-hand side for the inequalities. Use $b = []$ if there are no inequalities.
- [in] `B` (`double[][]`) – [optional] Constraint matrix for the equalities. Use $B = []$ if there are no equalities.
- [in] `c` (`double[]`) – [optional] Right-hand side for the equalities. Use $c = []$ if there are no equalities.
- [in] `l` (`double[]`) – [optional] Lower bounds on the variables. Use $-\infty$ to represent infinite lower bounds.
- [in] `u` (`double[]`) – [optional] Upper bounds on the variables. Use ∞ to represent infinite lower bounds.
- [in] `x0` (`double[]`) – [optional] An initial guess for the starting point. This information is ignored by **MOSEK**
- [in] `options` (`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:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|Cx-d\|_2^2 \\ \text{subject to} & x \geq 0. \end{array} \tag{15.3}$$

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

`Parameters`

- `[in] C (double[][])` – See problem (15.3).
- `[in] d (double[])` – See problem (15.3).
- `[in] x0 (double[])` – [optional] An initial guess for the starting point. This information is ignored by **MOSEK**
- `[in] options (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:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T Hx + f^T x \\ \text{subject to} & Ax \leq b, \\ & Bx = c, \\ & l \leq x \leq u. \end{array} \tag{15.4}$$

`Parameters`

- `[in] H (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.
- `[in] f (double[])` – See (15.4) for the definition.
- `[in] A (double[][])` – Constraint matrix for the inequalities. Use $A = []$ if there are no inequalities.
- `[in] b (double[])` – Right-hand side for the nequalities. Use $b = []$ if there are no inequalities.
- `[in] B (double[][])` – [optional] Constraint matrix for the equalities. Use $B = []$ if there are no equalities.
- `[in] c (double[])` – [optional] Right-hand side for the equalities. Use $c = []$ if there are no equalities.
- `[in] l (double[])` – [optional] Lower bounds on the variables. Use $-\infty$ to represent infinite lower bounds.
- `[in] u (double[])` – [optional] Upper bounds on the variables. Use ∞ to represent infinite upper bounds.
- `[in] x0 (double[])` – [optional] An initial guess for the starting point. This information is ignored by **MOSEK**
- `[in] options (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. $\frac{1}{2}x^T Hx + f^T x$.
- `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.

15.2 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*
- *duasen*
- *info*
- *symbcon*
- *callback*

15.2.1 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

$$\begin{array}{lrcccl} \text{minimize} & & & \sum_{j=1}^n c_j x_j + c^f & & \\ \text{subject to} & l_i^c & \leq & \sum_{j=1}^n a_{ij} x_j & \leq & u_i^c, \quad i = 1, \ldots, m, \\ & l_j^x & \leq & x_j & \leq & u_j^x, \quad j = 1, \ldots, n. \end{array}$$

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

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

where the conic constraint

$$x \in \mathcal{K} \tag{15.5}$$

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

$$\begin{array}{lrcccl} \text{minimize} & & & \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n q_{ij}^o x_i x_j + \sum_{j=1}^n c_j x_j + c^f & & \\ \text{subject to} & l_i^c & \leq & \frac{1}{2}\sum_{j=1}^n \sum_{k=1}^n q_{jk}^i x_j x_k + \sum_{j=1}^n a_{ij} x_j & \leq & u_i^c, \quad i = 1, \ldots, m, \\ & l_j^x & \leq & x & \leq & u_j^x, \quad j = 1, \ldots, n. \end{array}$$

The matrix

$$Q^o = \begin{bmatrix} q_{11}^o & \cdots & q_{1n}^o \\ \vdots & \ldots & \vdots \\ q_{n1}^0 & \cdots & q_{nn}^o \end{bmatrix}$$

must be symmetric positive semidefinite and the matrix

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

must be either symmetric negative semidefinite with the ith 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 ith 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.,

$$x_j \text{ integer-constrained for all } j \in | \tag{15.6}$$

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

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

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

- 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

$$l_i^c \leq \sum_{j=1}^n a_{ij} x_j + \sum_{j=1}^p \left\langle \overline{A}_{ij}, \overline{X}_j \right\rangle \leq u_i^c, \quad i = 1, \ldots, m,$$

for semide nite problems. The matrices are specifed in triplet format discard zero elements, and since they are symmetric only the lower triangulars should be speci ed.

l^c speci es the lower bounds of the constraints.

u^c speci es the upper bounds of the constraints.

l^x speci es the lower bounds on the variables x.

u^x speci es 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 eld 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, de ne

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

where K_t must have one of the following forms

- Free variables:

$$K_t = \{ x \in \mathbb{R}^{n^t} \}.$$

- Quadratic cones:

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

- Rotated quadratic cones:

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

The parameters of the optimization problem are stored using one or more sub elds of the `prob` structure using the naming convention in Table 15.1.

- `qcsubi` (`int[]`) – i subscript for element q^p_{ij} in Q^p. See (15.8)
- `qcsubj` (`double[]`) – j subscript for element q^p_{ij} in Q^p. See (15.8)
- `qcval` (`double[]`) – Numerical value for element q^p_{ij} in Q^p. See (15.8)
- `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. ∞ denotes an infinite upper bound. If the field is not defined or `buc==[]`, then all the upper bounds are assumed to be equal to ∞.
- `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. ∞ denotes an infinite upper bound. If the field is not defined or `bux==[]`, then all the upper bounds are assumed to be equal to ∞.
- `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 $\mathcal{I}$ in (15.6).
  - `.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 (15.5).
- `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 ith constraint.
- **var** (cell) – a cell array where names.var{j} contains the name of the jth variable.
- **barvar** (cell) – a cell array where names.barvar{j} contains the name of the jth semidefinite variable.
- **cone** (cell) – a cell array where names.cone{t} contains the name of the tth 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** (none) – 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

$$[\overline{C}_{\mathtt{barc.subj(t)}}]_{\mathtt{barc.subk(t)},\mathtt{barc.subl(t)}} = \mathtt{barc.val(t)}, \quad t = 1, 2, \ldots, \mathtt{length(barc.subj)}.$$

Only the lower triangular parts of $\overline{C}_j$ are specified, i.e., it is required that

$$\mathtt{barc.subk(t)} \geq \mathtt{barc.subl(t)}, \quad t = 1, 2, \ldots, \mathtt{length(barc.subk)},$$

and that

$$1 \leq \mathtt{barc.subk(t)} \leq \mathtt{bardim(barc.subj(t))}, \quad t = 1, 2, \ldots, \mathtt{length(barc.subj)}.,$$

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

$$[\overline{A}_{\mathtt{bara.subi(t)},\mathtt{bara.subj(t)}}]_{\mathtt{bara.subk(t)},\mathtt{bara.subl(t)}} = \mathtt{bara.val(t)}, \quad t = 1, 2, \ldots, \mathtt{length(bara.subi)}.$$

Only the lower triangular parts of $\overline{A}_{ij}$ are specified, i.e., it is required that

$$\mathtt{bara.subk(t)} \geq \mathtt{bara.subl(t)}, \quad t = 1, 2, \ldots, \mathtt{length(bara.subk)},$$

and that

$$1 \leq \mathtt{bara.subk(t)} \leq \mathtt{bardim(bara.subj(t))}, \quad t = 1, 2, \ldots, \mathtt{length(bara.subj)},$$

`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` (*`MSKprostae`*) – Problem status.
- `solsta` (*`MSKsolstae`*) – Solution status.
- `skc` (*`MSKstakeye`*) – Enumraint status keys.
- `skx` (*`MSKstakeye`*) – Variable status keys.
- `skn` (*`MSKstakeye`*) – Conic status keys.
- `xc` (`double[]`) – Constraint activities, i.e., $x_c = Ax$ where x is the optimal solution.
- `xx` (`double[]`) – The optimal x solution.

`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 *MSKdinfiteme* and *MSKiinfiteme* 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 Section 15.5 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.

15.3 Parameters

All parameters (alphabetical order)

- double parameters
- integer parameters
- string parameters

Parameters grouped by topic

Note: some parameters may appear in more than one group.

- Dual simplex optimizer
- Logging
- Analysis
- Basis identi cation
- Overall solver
- Data check
- Presolve

parameter means the central is followed very closely. On numerical unstable problems it may be worthwhile to increase this parameter.

`Accepted Values:` [0.0 ;0.9999]

`Default Value:` 1.0e-8

`Groups:` Interior-point method

`MSK_DPAR_INTPNT_TOL_PFEAS`
Primal feasibility tolerance used for linear and quadratic optimization problems.

`Accepted Values:` [0.0 ;1.0]

`Default Value:` 1.0e-8

`Groups:` Interior-point method , Termination criterion

`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.

`Accepted Values:` [1.0e-4 ;+inf]

`Default Value:` 1.0

`Groups:` Interior-point method

`MSK_DPAR_INTPNT_TOL_REL_GAP`
Relative gap termination tolerance.

`Accepted Values:` [1.0e-14 ;+inf]

`Default Value:` 1.0e-8

`Groups:` Termination criterion , Interior-point method

`MSK_DPAR_INTPNT_TOL_REL_STEP`
Relative step size to the boundary for linear and quadratic optimization problems.

`Accepted Values:` [1.0e-4 ;0.999999]

`Default Value:` 0.9999

`Groups:` Interior-point method

`MSK_DPAR_INTPNT_TOL_STEP_SIZE`
If the step size falls below the value of this parameter, then the interior-point optimizer assumes that it is stalled. In other words the interior-point optimizer does not make any progress and therefore it is better stop.

`Accepted Values:` [0.0 ;1.0]

`Default Value:` 1.0e-6

`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.

`Accepted Values:` [-inf ;+inf]

`Default Value:` -1.0e30

`Groups:` Termination criterion

`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 $-\infty$.

Accepted Values: [-inf ;+inf]

Default Value: -0.5e30

Groups: Termination criterion

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.

Accepted Values: [-inf ;+inf]

Default Value: -1.0

Groups: Mixed-integer optimization, Termination criterion

MSK_DPAR_MIO_MAX_TIME
This parameter limits the maximum time spent by the mixed-integer optimizer. A negative number means infinity.

Accepted Values: [-inf ;+inf]

Default Value: -1.0

Groups: Mixed-integer optimization, Termination criterion

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.

Accepted Values: [0.0 ;+inf]

Default Value: 0.0

Groups: Mixed-integer optimization

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.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e-3

Groups: Mixed-integer optimization, Termination criterion

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

Accepted Values: [1.0e-15 ;+inf]

Default Value: 1.0e-10

Groups: Mixed-integer optimization, Termination criterion

MSK_DPAR_MIO_TOL_ABS_GAP
Absolute optimality tolerance employed by the mixed-integer optimizer.

Accepted Values: [0.0 ;+inf]

Default Value: 0.0

Groups: Mixed-integer optimization

MSK_DPAR_MIO_TOL_ABS_RELAX_INT
Absolute relaxation tolerance of the integer constraints. I.e. $\min(|x| - \lfloor x \rfloor, \lceil x \rceil - |x|)$ is less than the tolerance then the integer restrictions assumed to be satisfied.

Accepted Values: [1e-9 ;+inf]

Default Value: 1.0e-5

Groups: Mixed-integer optimization

MSK_DPAR_MIO_TOL_FEAS
Feasibility tolerance for mixed integer solver.

Accepted Values: [1e-9 ;1e-3]

Default Value: 1.0e-6

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.

Accepted Values: [0.0 ;1.0]

Default Value: 0.0

Groups: Mixed-integer optimization

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

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e-4

Groups: Mixed-integer optimization, Termination criterion

MSK_DPAR_OPTIMIZER_MAX_TIME
Maximum amount of time the optimizer is allowed to spent on the optimization. A negative number means infinity.

Accepted Values: [-inf ;+inf]

Default Value: -1.0

Groups: Termination criterion

MSK_DPAR_PRESOLVE_TOL_ABS_LINDEP
Absolute tolerance employed by the linear dependency checker.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e-6

Groups: Presolve

MSK_DPAR_PRESOLVE_TOL_AIJ
Absolute zero tolerance employed for a_{ij} in the presolve.

Accepted Values: [1.0e-15 ;+inf]

Default Value: 1.0e-12

Groups: Presolve

MSK_DPAR_PRESOLVE_TOL_REL_LINDEP
Relative tolerance employed by the linear dependency checker.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e-10

Groups: Presolve

MSK_DPAR_PRESOLVE_TOL_S
Absolute zero tolerance employed for s_i in the presolve.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e-8

Groups: Presolve

MSK_DPAR_PRESOLVE_TOL_X
Absolute zero tolerance employed for x_j in the presolve.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e-8

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.

Accepted Values: [0 ;+inf]

Default Value: 1e-15

Groups: Interior-point method

MSK_DPAR_SEMIDEFINITE_TOL_APPROX
Tolerance to define a matrix to be positive semidefinite.

Accepted Values: [1.0e-15 ;+inf]

Default Value: 1.0e-10

Groups: Data check

MSK_DPAR_SIMPLEX_ABS_TOL_PIV
Absolute pivot tolerance employed by the simplex optimizers.

Accepted Values: [1.0e-12 ;+inf]

Default Value: 1.0e-7

Groups: Simplex optimizer

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.

Accepted Values: [1.0e-6 ;0.999999]

Default Value: 0.01

Groups: Basis identi cation , 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.

Accepted Values: [-inf ;+inf]

Default Value: 1.0e30

Groups: Termination criterion

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*.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: Interior-point method , Basis identi cation

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*.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: Interior-point method , Basis identi cation

MSK_IPAR_BI_MAX_ITERATIONS

Controls the maximum number of simplex iterations allowed to optimize a basis after the basis identification.

Accepted Values: [0 ;+inf]

Default Value: 1000000

Groups: Basis identi cation , Termination criterion

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.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: License manager

MSK_IPAR_CHECK_CONVEXITY

Specify the level of convexity check on quadratic problems

Accepted Values: *MSKcheckconvexitytypee*

Default Value: *MSK_CHECK_CONVEXITY_FULL*

Groups: Data check, Nonlinear convex method

MSK_IPAR_COMPRESS_STATFILE

Control compression of stat files.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

MSK_IPAR_INFEAS_GENERIC_NAMES

Controls whether generic names are used when an infeasible subproblem is created.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: Infeasibility report

`MSK_IPAR_LOG_SIM_MINOR`
Currently not in use.

`Accepted Values:` [0 ;+inf]

`Default Value:` 1

`Groups:` Simplex optimizer, Output information

`MSK_IPAR_LOG_STORAGE`
When turned on, **MOSEK** prints messages regarding the storage usage and allocation.

`Accepted Values:` [0 ;+inf]

`Default Value:` 0

`Groups:` Output information , Optimization 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.

`Accepted Values:` [-inf ;+inf]

`Default Value:` 10

`Groups:` Output information

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

`Accepted Values:` *MSKbranchdire*

`Default Value:` *MSK_BRANCH_DIR_FREE*

`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.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_OFF*

`Groups:` Mixed-integer optimization

`MSK_IPAR_MIO_CUT_CLIQUE`
Controls whether clique cuts should be generated.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_ON*

`Groups:` Mixed-integer optimization

`MSK_IPAR_MIO_CUT_CMIR`
Controls whether mixed integer rounding cuts should be generated.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_ON*

`Groups:` Mixed-integer optimization

`MSK_IPAR_MIO_CUT_GMI`
Controls whether GMI cuts should be generated.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_ON*

`Groups:` Mixed-integer optimization

`MSK_IPAR_MIO_CUT_IMPLIED_BOUND`
Controls whether implied bound cuts should be generated.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_OFF*

`Groups:` Mixed-integer optimization

`MSK_IPAR_MIO_CUT_KNAPSACK_COVER`
Controls whether knapsack cover cuts should be generated.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_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

`Accepted Values:` [-1 ;+1]

`Default Value:` -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.

`Accepted Values:` [-inf ;+inf]

`Default Value:` -1

`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.

`Accepted Values:` [-inf ;+inf]

`Default Value:` -1

`Groups:` Mixed-integer optimization, Termination criterion

`MSK_IPAR_MIO_MAX_NUM_RELAXS`
Maximum number of relaxations allowed during the branch and bound search. A negative value means infinite.

`Accepted Values:` [-inf ;+inf]

`Default Value:` -1

`Groups:` Mixed-integer optimization

`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.

`Accepted Values:` [-inf ;+inf]

Default Value: -1

Groups: Mixed-integer optimization, Termination criterion

MSK_IPAR_MIO_MODE
Controls whether the optimizer includes the integer restrictions when solving a (mixed) integer optimization problem.

Accepted Values: *MSKmiomodee*

Default Value: *MSK_MIO_MODE_SATISFIED*

Groups: Overall solver

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: Optimization system

MSK_IPAR_MIO_NODE_OPTIMIZER
Controls which optimizer is employed at the non-root nodes in the mixed-integer optimizer.

Accepted Values: *MSKoptimizertypee*

Default Value: *MSK_OPTIMIZER_FREE*

Groups: Mixed-integer optimization

MSK_IPAR_MIO_NODE_SELECTION
Controls the node selection strategy employed by the mixed-integer optimizer.

Accepted Values: *MSKmionodeseltypee*

Default Value: *MSK_MIO_NODE_SELECTION_FREE*

Groups: Mixed-integer optimization

MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE
Enables or disables perspective reformulation in presolve.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

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

Accepted Values: [-inf ;+inf]

Default Value: -1

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.

Accepted Values: [-1 ;+inf]

Default Value: -1

Groups: Mixed-integer optimization

MSK_IPAR_MIO_ROOT_OPTIMIZER
Controls which optimizer is employed at the root node in the mixed-integer optimizer.

Accepted Values: MSKoptimizertypee

Default Value: MSK_OPTIMIZER_FREE

Groups: Mixed-integer optimization

MSK_IPAR_MIO_ROOT_REPEAT_PRESOLVE_LEVEL
Controls whether presolve can be repeated at root node.

ˆ-1 The optimizer chooses whether presolve is repeated

ˆ0 Never repeat presolve

ˆ1 Always repeat presolve

Accepted Values: [-1 ;1]

Default Value: -1

Groups: Mixed-integer optimization

MSK_IPAR_MIO_VB_DETECTION_LEVEL
Controls how much e ort 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

Accepted Values: [-1 ;+2]

Default Value: -1

Groups: Mixed-integer optimization

MSK_IPAR_MT_SPINCOUNT
Set the number of iterations to spin before sleeping.

Accepted Values: [0 ;1000000000]

Default Value: 0

Groups: Optimization 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.

Accepted Values: [0 ;+inf]

Default Value: 0

Groups: Optimization system

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

Accepted Values: [0 ;+inf]

Default Value: 5

Groups: Data input/output

`MSK_IPAR_OPTIMIZER`
The parameter controls which optimizer is used to optimize the task.

Accepted Values: *MSKoptimizertypee*

Default Value: *MSK_OPTIMIZER_FREE*

Groups: Overall solver

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: Data input/output

`MSK_IPAR_PARAM_READ_IGN_ERROR`
If turned on, then errors in parameter settings is ignored.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_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 menas the parameter value is selected automatically.

Accepted Values: [-inf ;+inf]

Default Value: -1

Groups: Presolve

`MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES`
Control the maximum number of times the eliminator is tried.

Accepted Values: [-inf ;+inf]

Default Value: -1

Groups: Presolve

`MSK_IPAR_PRESOLVE_LEVEL`
Currently not used.

Accepted Values: [-inf ;+inf]

Default Value: -1

Groups: Overall solver, Presolve

`MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH`
The linear dependency check is potentially computationally expensive.

Accepted Values: [-inf ;+inf]

Default Value: 100

Groups: Presolve

`MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH`
The linear dependency check is potentially computationally expensive.

Accepted Values: [-inf ;+inf]

Default Value: 100

Groups: Presolve

`MSK_IPAR_PRESOLVE_LINDEP_USE`
Controls whether the linear constraints are checked for linear dependencies.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_ON*

`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.

`Accepted Values:` [-inf ;+inf]

`Default Value:` -1

`MSK_IPAR_PRESOLVE_USE`
Controls whether the presolve is applied to a problem before it is optimized.

`Accepted Values:` *MSKpresolvemodee*

`Default Value:` *MSK_PRESOLVE_MODE_FREE*

`Groups:` Overall solver, Presolve

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

`Accepted Values:` *MSKoptimizertypee*

`Default Value:` *MSK_OPTIMIZER_FREE*

`Groups:` Overall solver

`MSK_IPAR_READ_DATA_COMPRESSED`
If this option is turned on,it is assumed that the data file is compressed.

`Accepted Values:` *MSKcompresstypee*

`Default Value:` *MSK_COMPRESS_FREE*

`Groups:` Data input/output

`MSK_IPAR_READ_DATA_FORMAT`
Format of the data file to be read.

`Accepted Values:` *MSKdataformate*

`Default Value:` *MSK_DATA_FORMAT_EXTENSION*

`Groups:` Data input/output

`MSK_IPAR_READ_DEBUG`
Turns on additional debugging information when reading files.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_OFF*

`Groups:` Data input/output

`MSK_IPAR_READ_KEEP_FREE_CON`
Controls whether the free constraints are included in the problem.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_OFF*

`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 de ned for the rst time in the bounds section.

Accepted Values: MSKonoffkeye

Default Value: MSK_OFF

Groups: Data input/output

MSK_IPAR_READ_LP_QUOTED_NAMES
If a name is in quotes when reading an LP le, the quotes will be removed.

Accepted Values: MSKonoffkeye

Default Value: MSK_ON

Groups: Data input/output

MSK_IPAR_READ_MPS_FORMAT
Controls how strictly the MPS le reader interprets the MPS format.

Accepted Values: MSKmpsformate

Default Value: MSK_MPS_FORMAT_FREE

Groups: Data input/output

MSK_IPAR_READ_MPS_WIDTH
Controls the maximal number of characters allowed in one line of the MPS le.

Accepted Values: [80 ;+inf]

Default Value: 1024

Groups: Data input/output

MSK_IPAR_READ_TASK_IGNORE_PARAM
Controls whether MOSEK should ignore the parameter setting de ned in the task le and use the default parameter setting instead.

Accepted Values: MSKonoffkeye

Default Value: MSK_OFF

Groups: Data input/output

MSK_IPAR_SENSITIVITY_ALL
Not applicable.

Accepted Values: MSKonoffkeye

Default Value: MSK_OFF

Groups: Overall solver

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

Accepted Values: MSKoptimizertypee

Default Value: MSK_OPTIMIZER_FREE_SIMPLEX

Groups: Overall solver, Simplex optimizer

MSK_IPAR_SENSITIVITY_TYPE
Controls which type of sensitivity analysis is to be performed.

Accepted Values: MSKsensitivitytypee

Default Value: MSK_SENSITIVITY_TYPE_BASIS

Groups: Overall solver

`MSK_IPAR_SIM_BASIS_FACTOR_USE`

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

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_ON*

`Groups:` Simplex optimizer

`MSK_IPAR_SIM_DEGEN`

Controls how aggressively degeneration is handled.

`Accepted Values:` *MSKsimdegene*

`Default Value:` *MSK_SIM_DEGEN_FREE*

`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.

`Accepted Values:` [0 ;+inf]

`Default Value:` 90

`Groups:` Dual simplex optimizer

`MSK_IPAR_SIM_DUAL_PHASEONE_METHOD`

An experimental feature.

`Accepted Values:` [0 ;10]

`Default Value:` 0

`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.

`Accepted Values:` [0 ;100]

`Default Value:` 50

`Groups:` Dual simplex optimizer

`MSK_IPAR_SIM_DUAL_SELECTION`

Controls the choice of the incoming variable, known as the selection strategy, in the dual simplex optimizer.

`Accepted Values:` *MSKsimseltypee*

`Default Value:` *MSK_SIM_SELECTION_FREE*

`Groups:` Dual simplex optimizer

`MSK_IPAR_SIM_EXPLOIT_DUPVEC`

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

`Accepted Values:` *MSKsimdupvece*

`Default Value:` *MSK_SIM_EXPLOIT_DUPVEC_OFF*

Groups: Simplex optimizer

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

Accepted Values: *MSKsimhotstarte*

Default Value: *MSK_SIM_HOTSTART_FREE*

Groups: Simplex optimizer

MSK_IPAR_SIM_HOTSTART_LU
Determines if the simplex optimizer should exploit the initial factorization.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

MSK_IPAR_SIM_INTEGER
An experimental feature.

Accepted Values: [0 ;10]

Default Value: 0

Groups: Simplex optimizer

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

Accepted Values: [0 ;+inf]

Default Value: 10000000

Groups: Simplex optimizer, Termination criterion

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.

Accepted Values: [0 ;+inf]

Default Value: 250

Groups: Simplex optimizer

MSK_IPAR_SIM_NON_SINGULAR
Controls if the simplex optimizer ensures a non-singular basis, if possible.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

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.

Accepted Values: [0 ;+inf]

Default Value: 90

Groups: Primal simplex optimizer

MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD
An experimental feature.

Accepted Values: [0 ;10]

Default Value: 0

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.

Accepted Values: [0 ;100]

Default Value: 50

Groups: Primal simplex optimizer

MSK_IPAR_SIM_PRIMAL_SELECTION

Controls the choice of the incoming variable, known as the selection strategy, in the primal simplex optimizer.

Accepted Values: *MSKsimseltypee*

Default Value: *MSK_SIM_SELECTION_FREE*

Groups: Primal simplex optimizer

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.

Accepted Values: [0 ;+inf]

Default Value: 0

Groups: Simplex optimizer

MSK_IPAR_SIM_REFORMULATION

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

Accepted Values: *MSKsimreforme*

Default Value: *MSK_SIM_REFORMULATION_OFF*

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.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: Simplex optimizer

MSK_IPAR_SIM_SCALING

Controls how much effort is used in scaling the problem before a simplex optimizer is used.

Accepted Values: *MSKscalingtypee*

Default Value: *MSK_SCALING_FREE*

Groups: Simplex optimizer

MSK_IPAR_SIM_SCALING_METHOD

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

Accepted Values: *MSKscalingmethode*

Default Value: *MSK_SCALING_METHOD_POW2*

Groups: Simplex optimizer

MSK_IPAR_SIM_SOLVE_FORM
Controls whether the primal or the dual problem is solved by the primal-/dual-simplex optimizer.

Accepted Values: *MSKsolveforme*

Default Value: *MSK_SOLVE_FREE*

Groups: Simplex optimizer

MSK_IPAR_SIM_STABILITY_PRIORITY
Controls how high priority the numerical stability should be given.

Accepted Values: [0 ;100]

Default Value: 50

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).

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: Simplex optimizer

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: Progress call-back, Overall solver

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.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_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.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_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.

Accepted Values: [-inf ;+inf]

Default Value: -1

Groups: Data input/output , Solution input/output

MSK_IPAR_SOL_READ_WIDTH
Controls the maximal acceptable width of line in the solutions when read byMOSEK .

Accepted Values: [0 ;+inf]

Default Value: 1024

Groups: Data input/output , Solution input/output

MSK_IPAR_TIMING_LEVEL
Controls the a amount of timing performed inside MOSEK .

Accepted Values: [0 ;+inf]

Default Value: 1

Groups: Optimization system

MSK_IPAR_WRITE_BAS_CONSTRAINTS
Controls whether the constraint section is written to the basic solution le.

Accepted Values: MSKonoffkeye

Default Value: MSK_ON

Groups: Data input/output , Solution input/output

MSK_IPAR_WRITE_BAS_HEAD
Controls whether the header section is written to the basic solution le.

Accepted Values: MSKonoffkeye

Default Value: MSK_ON

Groups: Data input/output , Solution input/output

MSK_IPAR_WRITE_BAS_VARIABLES
Controls whether the variables section is written to the basic solution le.

Accepted Values: MSKonoffkeye

Default Value: MSK_ON

Groups: Data input/output , Solution input/output

MSK_IPAR_WRITE_DATA_COMPRESSED
Controls whether the data le is compressed while it is written. 0 means no compression while higher values mean more compression.

Accepted Values: [0 ;+inf]

Default Value: 0

Groups: Data input/output

MSK_IPAR_WRITE_DATA_FORMAT
Controls the le format when writing task data to a le.

Accepted Values: MSKdataformate

Default Value: MSK_DATA_FORMAT_EXTENSION

Groups: Data input/output

MSK_IPAR_WRITE_DATA_PARAM
If this option is turned on the parameter settings are written to the data le as parameters.

Accepted Values: MSKonoffkeye

Default Value: MSK_OFF

Groups: Data input/output

`MSK_IPAR_WRITE_FREE_CON`
Controls whether the free constraints are written to the data file.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: Data input/output

`MSK_IPAR_WRITE_GENERIC_NAMES`
Controls whether the generic names or user-defined names are used in the data file.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: Data input/output

`MSK_IPAR_WRITE_GENERIC_NAMES_IO`
Index origin used in generic names.

Accepted Values: [0 ;+inf]

Default Value: 1

Groups: Data input/output

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: Data input/output

`MSK_IPAR_WRITE_INT_CONSTRAINTS`
Controls whether the constraint section is written to the integer solution file.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: Data input/output , Solution input/output

`MSK_IPAR_WRITE_INT_HEAD`
Controls whether the header section is written to the integer solution file.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: Data input/output , Solution input/output

`MSK_IPAR_WRITE_INT_VARIABLES`
Controls whether the variables section is written to the integer solution file.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

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.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: Data input/output

`MSK_IPAR_WRITE_LP_LINE_WIDTH`
Maximum width of line in an LP file written by **MOSEK**.

`Accepted Values:` [40 ;+inf]

`Default Value:` 80

`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.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_ON*

`Groups:` Data input/output

`MSK_IPAR_WRITE_LP_STRICT_FORMAT`
Controls whether LP output files satisfy the LP format strictly.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_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.

`Accepted Values:` [0 ;+inf]

`Default Value:` 10

`Groups:` Data input/output

`MSK_IPAR_WRITE_MPS_FORMAT`
Controls in which format the MPS is written.

`Accepted Values:` *MSKmpsformate*

`Default Value:` *MSK_MPS_FORMAT_FREE*

`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.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_ON*

`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.

`Accepted Values:` [0 ;+inf]

`Default Value:` 15

`Groups:` Data input/output

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

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_ON*

`Groups:` Data input/output , Solution input/output

`MSK_SPAR_DEBUG_FILE_NAME`
MOSEK debug file.

`Accepted Values:` Any valid file name.

`Groups:` Data input/output

`MSK_SPAR_INT_SOL_FILE_NAME`
Name of the `int` solution file.

`Accepted Values:` Any valid file name.

`Groups:` Data input/output , Solution input/output

`MSK_SPAR_ITR_SOL_FILE_NAME`
Name of the `itr` solution file.

`Accepted Values:` Any valid file name.

`Groups:` Data input/output , Solution input/output

`MSK_SPAR_MIO_DEBUG_STRING`
For internal use only.

`Accepted Values:` 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.

`Accepted Values:` Any valid string.

`Default Value:` %%

`Groups:` Data input/output

`MSK_SPAR_PARAM_READ_FILE_NAME`
Modifications to the parameter database is read from this file.

`Accepted Values:` Any valid file name.

`Groups:` Data input/output

`MSK_SPAR_PARAM_WRITE_FILE_NAME`
The parameter database is written to this file.

`Accepted Values:` 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 Values:` 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 Values:` 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 Values:` 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 Values:` 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 Values:` Any valid string.

`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 Values:` Any valid string.

`Groups:` Data input/output

`MSK_SPAR_SENSITIVITY_RES_FILE_NAME`
`Accepted Values:` 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 Values:` 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 Values:` 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 Values:` 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 Values:` Any valid file name.

`Groups:` Data input/output , Solution input/output

`MSK_SPAR_STAT_FILE_NAME`
Statistics file name.

`Accepted Values:` Any valid file name.

`Groups:` Data input/output

MSK_SPAR_STAT_KEY
Key used when writing the summary le.

Accepted Values: Any valid XML string.

Groups: Data input/output

MSK_SPAR_STAT_NAME
Name used when writing the statistics le.

Accepted Values: Any valid XML string.

Groups: Data input/output

MSK_SPAR_WRITE_LP_GEN_VAR_NAME
Sometimes when an LP le is written additional variables must be inserted. They will have the pre x denoted by this parameter.

Accepted Values: Any valid string.

Default Value: xmskgen

Groups: Data input/output

15.3.2 Dual simplex optimizer parameters.

- MSK_IPAR_SIM_DUAL_CRASH
- MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION
- MSK_IPAR_SIM_DUAL_SELECTION

15.3.3 Logging parameters.

- 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_FACTOR
- MSK_IPAR_LOG_FEAS_REPAIR
- MSK_IPAR_LOG_FILE
- MSK_IPAR_LOG_HEAD
- MSK_IPAR_LOG_INFEAS_ANA
- MSK_IPAR_LOG_INTPNT
- MSK_IPAR_LOG_MIO
- MSK_IPAR_LOG_MIO_FREQ
- MSK_IPAR_LOG_OPTIMIZER
- 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*

15.3.4 Analysis parameters.

- *MSK_IPAR_ANA_SOL_BASIS*
- *MSK_DPAR_ANA_SOL_INFEAS_TOL*
- *MSK_IPAR_ANA_SOL_PRINT_VIOLATED*
- *MSK_IPAR_LOG_ANA_PRO*

15.3.5 Basis identification parameters.

- *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*
- *MSK_DPAR_SIM_LU_TOL_REL_PIV*

15.3.6 Overall solver parameters.

- *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_USE*
- *MSK_IPAR_PRIMAL_REPAIR_OPTIMIZER*
- *MSK_IPAR_SENSITIVITY_ALL*
- *MSK_IPAR_SENSITIVITY_OPTIMIZER*
- *MSK_IPAR_SENSITIVITY_TYPE*
- *MSK_IPAR_SOLUTION_CALLBACK*

15.3.7 Data check parameters.

- *MSK_IPAR_CHECK_CONVEXITY*
- *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_IPAR_LOG_CHECK_CONVEXITY*
- *MSK_DPAR_SEMIDEFINITE_TOL_APPROX*

15.3.8 Presolve parameters.

- *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_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_USE*

15.3.9 Termination criterion parameters.

- *MSK_DPAR_BASIS_REL_TOL_S*
- *MSK_DPAR_BASIS_TOL_S*
- *MSK_DPAR_BASIS_TOL_X*
- *MSK_IPAR_BI_MAX_ITERATIONS*
- *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_IPAR_INTPNT_MAX_ITERATIONS*
- *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_IPAR_MIO_MAX_NUM_BRANCHES*
- *MSK_IPAR_MIO_MAX_NUM_SOLUTIONS*
- *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_IPAR_SIM_MAX_ITERATIONS*
- *MSK_DPAR_UPPER_OBJ_CUT*
- *MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH*

15.3.10 Infeasibility report parameters.

- *MSK_IPAR_INFEAS_GENERIC_NAMES*
- *MSK_IPAR_INFEAS_REPORT_LEVEL*
- *MSK_IPAR_LOG_INFEAS_ANA*

15.3.11 Data input/output parameters.

- *MSK_SPAR_BAS_SOL_FILE_NAME*
- *MSK_SPAR_DATA_FILE_NAME*
- *MSK_SPAR_DEBUG_FILE_NAME*
- *MSK_IPAR_INFEAS_REPORT_AUTO*
- *MSK_SPAR_INT_SOL_FILE_NAME*
- *MSK_SPAR_ITR_SOL_FILE_NAME*
- *MSK_IPAR_LOG_FILE*
- *MSK_SPAR_MIO_DEBUG_STRING*
- *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_SPAR_PARAM_COMMENT_SIGN*
- *MSK_IPAR_PARAM_READ_CASE_NAME*
- *MSK_SPAR_PARAM_READ_FILE_NAME*
- *MSK_IPAR_PARAM_READ_IGN_ERROR*
- *MSK_SPAR_PARAM_WRITE_FILE_NAME*
- *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_SPAR_READ_MPS_BOU_NAME*
- *MSK_IPAR_READ_MPS_FORMAT*
- *MSK_SPAR_READ_MPS_OBJ_NAME*
- *MSK_SPAR_READ_MPS_RAN_NAME*
- *MSK_SPAR_READ_MPS_RHS_NAME*
- *MSK_IPAR_READ_MPS_WIDTH*
- *MSK_IPAR_READ_TASK_IGNORE_PARAM*
- *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_IPAR_SOL_READ_NAME_WIDTH*
- *MSK_IPAR_SOL_READ_WIDTH*
- *MSK_SPAR_STAT_FILE_NAME*
- *MSK_SPAR_STAT_KEY*
- *MSK_SPAR_STAT_NAME*
- *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_SPAR_WRITE_LP_GEN_VAR_NAME*
- *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*

15.3.12 Interior-point method parameters.

- *MSK_IPAR_BI_IGNORE_MAX_ITER*
- *MSK_IPAR_BI_IGNORE_NUM_ERROR*
- *MSK_DPAR_CHECK_CONVEXITY_REL_TOL*
- *MSK_IPAR_INTPNT_BASIS*
- *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_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_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_IPAR_INTPNT_OFF_COL_TRH*
- *MSK_IPAR_INTPNT_ORDER_METHOD*
- *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_IPAR_INTPNT_REGULARIZATION_USE*
- *MSK_IPAR_INTPNT_SCALING*
- *MSK_IPAR_INTPNT_SOLVE_FORM*
- *MSK_IPAR_INTPNT_STARTING_POINT*
- *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_IPAR_LOG_INTPNT
- MSK_IPAR_LOG_PRESOLVE
- MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL

15.3.13 Debugging parameters.

- MSK_IPAR_AUTO_SORT_A_BEFORE_OPT

15.3.14 Solution input/output parameters.

- MSK_SPAR_BAS_SOL_FILE_NAME
- MSK_IPAR_INFEAS_REPORT_AUTO
- MSK_SPAR_INT_SOL_FILE_NAME
- MSK_SPAR_ITR_SOL_FILE_NAME
- MSK_IPAR_SOL_FILTER_KEEP_BASIC
- MSK_IPAR_SOL_FILTER_KEEP_RANGED
- 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_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

15.3.15 Progress call-back parameters.

- *MSK_IPAR_SOLUTION_CALLBACK*

15.3.16 Conic interior-point method parameters.

- *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*

15.3.17 Simplex optimizer parameters.

- *MSK_DPAR_BASIS_REL_TOL_S*
- *MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE*
- *MSK_DPAR_BASIS_TOL_S*
- *MSK_DPAR_BASIS_TOL_X*
- *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_INTEGER*
- *MSK_DPAR_SIM_LU_TOL_REL_PIV*
- *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*
- *MSK_DPAR_SIMPLEX_ABS_TOL_PIV*

15.3.18 Primal simplex optimizer parameters.

- *MSK_IPAR_SIM_PRIMAL_CRASH*
- *MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION*
- *MSK_IPAR_SIM_PRIMAL_SELECTION*

15.3.19 Mixed-integer optimization parameters.

- *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_DPAR_MIO_DISABLE_TERM_TIME*
- *MSK_IPAR_MIO_HEURISTIC_LEVEL*
- *MSK_IPAR_MIO_MAX_NUM_BRANCHES*
- *MSK_IPAR_MIO_MAX_NUM_RELAXS*
- *MSK_IPAR_MIO_MAX_NUM_SOLUTIONS*
- *MSK_DPAR_MIO_MAX_TIME*
- *MSK_DPAR_MIO_NEAR_TOL_ABS_GAP*
- *MSK_DPAR_MIO_NEAR_TOL_REL_GAP*
- *MSK_IPAR_MIO_NODE_OPTIMIZER*
- *MSK_IPAR_MIO_NODE_SELECTION*
- *MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE*
- *MSK_IPAR_MIO_PROBING_LEVEL*
- *MSK_DPAR_MIO_REL_GAP_CONST*
- *MSK_IPAR_MIO_RINS_MAX_NODES*
- *MSK_IPAR_MIO_ROOT_OPTIMIZER*
- *MSK_IPAR_MIO_ROOT_REPEAT_PRESOLVE_LEVEL*
- *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_MIO_VB_DETECTION_LEVEL*

15.3.20 Optimization system parameters.

- *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_TIMING_LEVEL*

15.3.21 License manager parameters.

- *MSK_IPAR_CACHE_LICENSE*
- *MSK_IPAR_LICENSE_DEBUG*
- *MSK_IPAR_LICENSE_PAUSE_TIME*
- *MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS*
- *MSK_IPAR_LICENSE_WAIT*

15.3.22 Output information parameters.

- *MSK_IPAR_INFEAS_REPORT_LEVEL*
- *MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS*
- *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_FACTOR*
- *MSK_IPAR_LOG_FEAS_REPAIR*
- *MSK_IPAR_LOG_FILE*
- *MSK_IPAR_LOG_HEAD*
- *MSK_IPAR_LOG_INFEAS_ANA*
- *MSK_IPAR_LOG_INTPNT*
- *MSK_IPAR_LOG_MIO*
- *MSK_IPAR_LOG_MIO_FREQ*
- *MSK_IPAR_LOG_OPTIMIZER*

- *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*

15.3.23 Nonlinear convex method parameters.

- *MSK_IPAR_CHECK_CONVEXITY*
- *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_LOG_CHECK_CONVEXITY*

15.4 Response codes

- Termination codes
- Error codes
- Warning codes

15.4.1 Termination Codes

`MSK_RES_OK (0)`
No error occurred.

`MSK_RES_TRM_INTERNAL (10030)`
The optimizer terminated due to some internal reason. Please contact **MOSEK** support.

`MSK_RES_TRM_INTERNAL_STOP (10031)`
The optimizer terminated for internal reasons. Please contact **MOSEK** support.

`MSK_RES_TRM_MAX_ITERATIONS (10000)`
The optimizer terminated at the maximum number of iterations.

`MSK_RES_TRM_MAX_NUM_SETBACKS (10020)`
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_MAX_TIME (10001)`
The optimizer terminated at the maximum amount of time.

`MSK_RES_TRM_MIO_NEAR_ABS_GAP (10004)`
The mixed-integer optimizer terminated because the near optimal absolute gap tolerance was satisfied.

`MSK_RES_TRM_MIO_NEAR_REL_GAP (10003)`
The mixed-integer optimizer terminated because the near optimal relative gap tolerance was satisfied.

`MSK_RES_TRM_MIO_NUM_BRANCHES (10009)`
The mixed-integer optimizer terminated as to the maximum number of branches was reached.

`MSK_RES_TRM_MIO_NUM_RELAXS (10008)`
The mixed-integer optimizer terminated as the maximum number of relaxations was reached.

`MSK_RES_TRM_NUM_MAX_NUM_INT_SOLUTIONS (10015)`
The mixed-integer optimizer terminated as the maximum number of feasible solutions was reached.

`MSK_RES_TRM_NUMERICAL_PROBLEM (10025)`
The optimizer terminated due to numerical problems.

`MSK_RES_TRM_OBJECTIVE_RANGE (10002)`
The optimizer terminated on the bound of the objective range.

`MSK_RES_TRM_STALL (10006)`
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 (10007)`
The optimizer terminated due to the return of the user-defined call-back function.

15.4.2 Error Codes

`MSK_RES_ERR_AD_INVALID_CODELIST (3102)`
The code list data was invalid.

`MSK_RES_ERR_API_ARRAY_TOO_SMALL (3001)`
An input array was too short.

`MSK_RES_ERR_API_CB_CONNECT (3002)`
Failed to connect a callback object.

`MSK_RES_ERR_API_FATAL_ERROR (3005)`
An internal error occurred in the API. Please report this problem.

`MSK_RES_ERR_API_INTERNAL (3999)`
An internal fatal error occurred in an interface function.

`MSK_RES_ERR_ARG_IS_TOO_LARGE (1227)`
The value of a argument is too small.

`MSK_RES_ERR_ARG_IS_TOO_SMALL (1226)`
The value of a argument is too small.

`MSK_RES_ERR_ARGUMENT_DIMENSION (1201)`
A function argument is of incorrect dimension.

`MSK_RES_ERR_ARGUMENT_IS_TOO_LARGE (5005)`
The value of a function argument is too large.

`MSK_RES_ERR_ARGUMENT_LENNEQ (1197)`
Incorrect length of arguments.

`MSK_RES_ERR_ARGUMENT_PERM_ARRAY (1299)`
An invalid permutation array is specified.

`MSK_RES_ERR_ARGUMENT_TYPE (1198)`
Incorrect argument type.

`MSK_RES_ERR_BAR_VAR_DIM (3920)`
The dimension of a symmetric matrix variable has to greater than 0.

`MSK_RES_ERR_BASIS (1266)`
An invalid basis is specified. Either too many or too few basis variables are specified.

`MSK_RES_ERR_BASIS_FACTOR (1610)`
The factorization of the basis is invalid.

`MSK_RES_ERR_BASIS_SINGULAR (1615)`
The basis is singular and hence cannot be factored.

`MSK_RES_ERR_BLANK_NAME (1070)`
An all blank name has been specified.

`MSK_RES_ERR_CANNOT_CLONE_NL (2505)`
A task with a nonlinear function call-back cannot be cloned.

`MSK_RES_ERR_CANNOT_HANDLE_NL (2506)`
A function cannot handle a task with nonlinear function call-backs.

`MSK_RES_ERR_CBF_DUPLICATE_ACOORD (7116)`
Duplicate index in ACOORD.

`MSK_RES_ERR_CBF_DUPLICATE_BCOORD (7115)`
Duplicate index in BCOORD.

`MSK_RES_ERR_CBF_DUPLICATE_CON (7108)`
Duplicate CON keyword.

`MSK_RES_ERR_CBF_DUPLICATE_INT (7110)`
Duplicate INT keyword.

`MSK_RES_ERR_CBF_DUPLICATE_OBJ (7107)`
Duplicate OBJ keyword.

`MSK_RES_ERR_CBF_DUPLICATE_OBJACOORD (7114)`
Duplicate index in OBJCOORD.

`MSK_RES_ERR_CBF_DUPLICATE_VAR (7109)`
Duplicate VAR keyword.

`MSK_RES_ERR_CBF_INVALID_CON_TYPE (7112)`
Invalid constraint type.

`MSK_RES_ERR_CBF_INVALID_DOMAIN_DIMENSION (7113)`
Invalid domain dimension.

`MSK_RES_ERR_CONE_SIZE (1301)`
A cone with too few members is specified.

`MSK_RES_ERR_CONE_TYPE (1305)`
Invalid cone type specified.

`MSK_RES_ERR_CONE_TYPE_STR (1306)`
Invalid cone type specified.

`MSK_RES_ERR_DATA_FILE_EXT (1055)`
The data file format cannot be determined from the file name.

`MSK_RES_ERR_DUP_NAME (1071)`
The same name was used multiple times for the same problem item type.

`MSK_RES_ERR_DUPLICATE_AIJ (1385)`
An element in the A matrix is specified twice.

`MSK_RES_ERR_DUPLICATE_BARVARIABLE_NAMES (4502)`
Two barvariable names are identical.

`MSK_RES_ERR_DUPLICATE_CONE_NAMES (4503)`
Two cone names are identical.

`MSK_RES_ERR_DUPLICATE_CONSTRAINT_NAMES (4500)`
Two constraint names are identical.

`MSK_RES_ERR_DUPLICATE_VARIABLE_NAMES (4501)`
Two variable names are identical.

`MSK_RES_ERR_END_OF_FILE (1059)`
End of file reached.

`MSK_RES_ERR_FACTOR (1650)`
An error occurred while factorizing a matrix.

`MSK_RES_ERR_FEASREPAIR_CANNOT_RELAX (1700)`
An optimization problem cannot be relaxed. This is the case e.g. for general nonlinear optimization problems.

`MSK_RES_ERR_FEASREPAIR_INCONSISTENT_BOUND (1702)`
The upper bound is less than the lower bound for a variable or a constraint. Please correct this before running the feasibility repair.

`MSK_RES_ERR_FEASREPAIR_SOLVING_RELAXED (1701)`
The relaxed problem could not be solved to optimality. Please consult the log file for further details.

`MSK_RES_ERR_FILE_LICENSE (1007)`
Invalid license file.

`MSK_RES_ERR_FILE_OPEN (1052)`
Error while opening a file.

`MSK_RES_ERR_FILE_READ (1053)`
File read error.

`MSK_RES_ERR_FILE_WRITE (1054)`
File write error.

`MSK_RES_ERR_FIRST (1261)`
Invalid `first`.

`MSK_RES_ERR_FIRSTI (1285)`
Invalid `firsti`.

`MSK_RES_ERR_FIRSTJ (1287)`
Invalid `firstj`.

`MSK_RES_ERR_FIXED_BOUND_VALUES (1425)`
A fixed constraint/variable has been specified using the bound keys but the numerical value of the lower and upper bound is different.

`MSK_RES_ERR_FLEXLM (1014)`
The FLEXlm license manager reported an error.

`MSK_RES_ERR_GLOBAL_INV_CONIC_PROBLEM (1503)`
The global optimizer can only be applied to problems without semidefinite variables.

`MSK_RES_ERR_HUGE_AIJ (1380)`
A numerically huge value is specified for an $a_{i,j}$ element in A. The parameter *MSK_DPAR_DATA_TOL_AIJ_HUGE* controls when an $a_{i,j}$ is considered huge.

`MSK_RES_ERR_HUGE_C (1375)`
A huge value in absolute size is specified for one c_j.

`MSK_RES_ERR_IDENTICAL_TASKS (3101)`
Some tasks related to this function call were identical. Unique tasks were expected.

`MSK_RES_ERR_IN_ARGUMENT (1200)`
A function argument is incorrect.

`MSK_RES_ERR_INDEX (1235)`
An index is out of range.

`MSK_RES_ERR_INDEX_ARR_IS_TOO_LARGE (1222)`
An index in an array argument is too large.

`MSK_RES_ERR_INDEX_ARR_IS_TOO_SMALL (1221)`
An index in an array argument is too small.

`MSK_RES_ERR_INDEX_IS_TOO_LARGE (1204)`
An index in an argument is too large.

`MSK_RES_ERR_INDEX_IS_TOO_SMALL (1203)`
An index in an argument is too small.

`MSK_RES_ERR_INF_DOU_INDEX (1219)`
A double information index is out of range for the specified type.

`MSK_RES_ERR_INF_DOU_NAME (1230)`
A double information name is invalid.

`MSK_RES_ERR_INF_INT_INDEX (1220)`
An integer information index is out of range for the specified type.

`MSK_RES_ERR_INF_INT_NAME (1231)`
An integer information name is invalid.

`MSK_RES_ERR_INF_LINT_INDEX (1225)`
A long integer information index is out of range for the specified type.

`MSK_RES_ERR_INF_LINT_NAME (1234)`
A long integer information name is invalid.

`MSK_RES_ERR_INF_TYPE (1232)`
The information type is invalid.

`MSK_RES_ERR_INFEAS_UNDEFINED (3910)`
The requested value is not defined for this solution type.

`MSK_RES_ERR_INFINITE_BOUND (1400)`
A numerically huge bound value is specified.

`MSK_RES_ERR_INT64_TO_INT32_CAST (3800)`
An 32 bit integer could not cast to a 64 bit integer.

`MSK_RES_ERR_INV_SK_STR (1269)`
Invalid status key string encountered.

`MSK_RES_ERR_INV_SKC (1267)`
Invalid value in `skc`.

`MSK_RES_ERR_INV_SKN (1274)`
Invalid value in `skn`.

`MSK_RES_ERR_INV_SKX (1268)`
Invalid value in `skx`.

`MSK_RES_ERR_INV_VAR_TYPE (1258)`
An invalid variable type is specified for a variable.

`MSK_RES_ERR_INVALID_ACCMODE (2520)`
An invalid access mode is specified.

`MSK_RES_ERR_INVALID_AIJ (1473)`
$a_{i,j}$ contains an invalid floating point value, i.e. a `NaN` or an infinite value.

`MSK_RES_ERR_INVALID_AMPL_STUB (3700)`
Invalid AMPL stub.

`MSK_RES_ERR_INVALID_BARVAR_NAME (1079)`
An invalid symmetric matrix variable name is used.

`MSK_RES_ERR_INVALID_COMPRESSION (1800)`
Invalid compression type.

`MSK_RES_ERR_INVALID_CON_NAME (1076)`
An invalid constraint name is used.

`MSK_RES_ERR_INVALID_CONE_NAME (1078)`
An invalid cone name is used.

`MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_CONES (4005)`
The file format does not support a problem with conic constraints.

`MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_GENERAL_NL (4010)`
The file format does not support a problem with general nonlinear terms.

`MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_SYM_MAT (4000)`
The file format does not support a problem with symmetric matrix variables.

`MSK_RES_ERR_INVALID_FILE_NAME (1056)`
An invalid file name has been specified.

`MSK_RES_ERR_INVALID_FORMAT_TYPE (1283)`
Invalid format type.

`MSK_RES_ERR_INVALID_IDX (1246)`
A specified index is invalid.

`MSK_RES_ERR_INVALID_IOMODE (1801)`
Invalid io mode.

`MSK_RES_ERR_INVALID_MAX_NUM (1247)`
A specified index is invalid.

`MSK_RES_ERR_INVALID_NAME_IN_SOL_FILE (1170)`
An invalid name occurred in a solution file.

`MSK_RES_ERR_INVALID_OBJ_NAME (1075)`
An invalid objective name is specified.

`MSK_RES_ERR_INVALID_OBJECTIVE_SENSE (1445)`
An invalid objective sense is specified.

`MSK_RES_ERR_INVALID_PROBLEM_TYPE (6000)`
An invalid problem type.

`MSK_RES_ERR_INVALID_SOL_FILE_NAME (1057)`
An invalid file name has been specified.

`MSK_RES_ERR_INVALID_STREAM (1062)`
An invalid stream is referenced.

`MSK_RES_ERR_INVALID_SURPLUS (1275)`
Invalid surplus.

`MSK_RES_ERR_INVALID_SYM_MAT_DIM (3950)`
A sparse symmetric matrix of invalid dimension is specified.

`MSK_RES_ERR_INVALID_TASK (1064)`
The `task` is invalid.

`MSK_RES_ERR_INVALID_UTF8 (2900)`
An invalid UTF8 string is encountered.

`MSK_RES_ERR_INVALID_VAR_NAME (1077)`
An invalid variable name is used.

`MSK_RES_ERR_INVALID_WCHAR (2901)`
An invalid `wchar` string is encountered.

`MSK_RES_ERR_INVALID_WHICHSOL (1228)`
`whichsol` is invalid.

`MSK_RES_ERR_JSON_DATA (1179)`
Inconsistent data in JSON Task file

`MSK_RES_ERR_JSON_FORMAT (1178)`
Error in an JSON Task file

`MSK_RES_ERR_JSON_MISSING_DATA (1180)`
Missing data section in JSON task file.

`MSK_RES_ERR_JSON_NUMBER_OVERFLOW (1177)`
Invalid number entry - wrong type or value overflow.

`MSK_RES_ERR_JSON_STRING (1176)`
Error in JSON string.

`MSK_RES_ERR_JSON_SYNTAX (1175)`
Syntax error in an JSON data

`MSK_RES_ERR_LAST (1262)`
Invalid index `last`. A given index was out of expected range.

`MSK_RES_ERR_LASTI (1286)`
Invalid `lasti`.

`MSK_RES_ERR_LASTJ (1288)`
Invalid `lastj`.

`MSK_RES_ERR_LAU_ARG_K (7012)`
Invalid argument k.

`MSK_RES_ERR_LAU_ARG_M (7010)`
Invalid argument m.

`MSK_RES_ERR_LAU_ARG_N (7011)`
Invalid argument n.

`MSK_RES_ERR_LAU_ARG_TRANS (7018)`
Invalid argument trans.

`MSK_RES_ERR_LAU_ARG_TRANSA (7015)`
Invalid argument transa.

`MSK_RES_ERR_LAU_ARG_TRANSB (7016)`
Invalid argument transb.

`MSK_RES_ERR_LAU_ARG_UPLO (7017)`
Invalid argument uplo.

`MSK_RES_ERR_LAU_INVALID_LOWER_TRIANGULAR_MATRIX (7002)`
An invalid lower triangular matrix.

`MSK_RES_ERR_LAU_INVALID_SPARSE_SYMMETRIC_MATRIX (7019)`
An invalid sparse symmetric matrix is specfified. Note only the lower triangular part with no duplicates is specifed.

`MSK_RES_ERR_LAU_NOT_POSITIVE_DEFINITE (7001)`
A matrix is not positive definite.

`MSK_RES_ERR_LAU_SINGULAR_MATRIX (7000)`
A matrix is singular.

`MSK_RES_ERR_LAU_UNKNOWN (7005)`
An unknown error.

`MSK_RES_ERR_LICENSE (1000)`
Invalid license.

`MSK_RES_ERR_LICENSE_CANNOT_ALLOCATE (1020)`
The license system cannot allocate the memory required.

`MSK_RES_ERR_LICENSE_CANNOT_CONNECT (1021)`
MOSEK cannot connect to the license server. Most likely the license server is not up and running.

`MSK_RES_ERR_LICENSE_EXPIRED (1001)`
The license has expired.

`MSK_RES_ERR_LICENSE_FEATURE (1018)`
A requested feature is not available in the license file(s). Most likely due to an incorrect license system setup.

`MSK_RES_ERR_LICENSE_INVALID_HOSTID (1025)`
The host ID specified in the license file does not match the host ID of the computer.

`MSK_RES_ERR_LICENSE_MAX (1016)`
Maximum number of licenses is reached.

`MSK_RES_ERR_LICENSE_MOSEKLM_DAEMON (1017)`
The MOSEKLM license manager daemon is not up and running.

`MSK_RES_ERR_LICENSE_NO_SERVER_LINE (1028)`
There is no `SERVER` line in the license file. All non-zero license count features need at least one `SERVER` line.

`MSK_RES_ERR_LICENSE_NO_SERVER_SUPPORT (1027)`
The license server does not support the requested feature. Possible reasons for this error include:

- The feature has expired.
- The feature's start date is later than today's date.
- The version requested is higher than feature's the highest supported version.
- A corrupted license file.

Try restarting the license and inspect the license server debug file, usually called `lmgrd.log`.

`MSK_RES_ERR_LICENSE_SERVER (1015)`
The license server is not responding.

`MSK_RES_ERR_LICENSE_SERVER_VERSION (1026)`
The version specified in the checkout request is greater than the highest version number the daemon supports.

`MSK_RES_ERR_LICENSE_VERSION (1002)`
The license is valid for another version of **MOSEK**.

`MSK_RES_ERR_LINK_FILE_DLL (1040)`
A file cannot be linked to a stream in the DLL version.

`MSK_RES_ERR_LIVING_TASKS (1066)`
All tasks associated with an enviroment must be deleted before the environment is deleted. There are still some undeleted tasks.

`MSK_RES_ERR_LOWER_BOUND_IS_A_NAN (1390)`
The lower bound specified is not a number (nan).

`MSK_RES_ERR_LP_DUP_SLACK_NAME (1152)`
The name of the slack variable added to a ranged constraint already exists.

`MSK_RES_ERR_LP_EMPTY (1151)`
The problem cannot be written to an LP formatted file.

`MSK_RES_ERR_LP_FILE_FORMAT (1157)`
Syntax error in an LP file.

`MSK_RES_ERR_LP_FORMAT (1160)`
Syntax error in an LP file.

`MSK_RES_ERR_LP_FREE_CONSTRAINT (1155)`
Free constraints cannot be written in LP file format.

`MSK_RES_ERR_LP_INCOMPATIBLE (1150)`
The problem cannot be written to an LP formatted file.

`MSK_RES_ERR_LP_INVALID_CON_NAME (1171)`
A constraint name is invalid when used in an LP formatted file.

`MSK_RES_ERR_LP_INVALID_VAR_NAME (1154)`
A variable name is invalid when used in an LP formatted file.

`MSK_RES_ERR_LP_WRITE_CONIC_PROBLEM (1163)`
The problem contains cones that cannot be written to an LP formatted file.

`MSK_RES_ERR_LP_WRITE_GECO_PROBLEM (1164)`
The problem contains general convex terms that cannot be written to an LP formatted file.

`MSK_RES_ERR_LU_MAX_NUM_TRIES (2800)`
Could not compute the LU factors of the matrix within the maximum number of allowed tries.

`MSK_RES_ERR_MAX_LEN_IS_TOO_SMALL (1289)`
An maximum length that is too small has been specified.

`MSK_RES_ERR_MAXNUMBARVAR (1242)`
The maximum number of semidefinite variables specified is smaller than the number of semidefinite variables in the task.

`MSK_RES_ERR_MAXNUMCON (1240)`
The maximum number of constraints specified is smaller than the number of constraints in the task.

`MSK_RES_ERR_MAXNUMCONE (1304)`
The value specified for `maxnumcone` is too small.

`MSK_RES_ERR_MAXNUMQNZ (1243)`
The maximum number of non-zeros specified for the Q matrices is smaller than the number of non-zeros in the current Q matrices.

MSK_RES_ERR_MAXNUMVAR (1241)
The maximum number of variables speci ed is smaller than the number of variables in the task.

MSK_RES_ERR_MIO_INTERNAL (5010)
A fatal error occurred in the mixed integer optimizer. Please contact MOSEK support.

MSK_RES_ERR_MIO_INVALID_NODE_OPTIMIZER (7131)
An invalid node optimizer was selected for the problem type.

MSK_RES_ERR_MIO_INVALID_ROOT_OPTIMIZER (7130)
An invalid root optimizer was selected for the problem type.

MSK_RES_ERR_MIO_NO_OPTIMIZER (1551)
No optimizer is available for the current class of integer optimization problems.

MSK_RES_ERR_MIO_NOT_LOADED (1553)
The mixed-integer optimizer is not loaded.

MSK_RES_ERR_MISSING_LICENSE_FILE (1008)
MOSEK cannot license le or a token server. See the MOSEK installation manual for details.

MSK_RES_ERR_MIXED_CONIC_AND_NL (1501)
The problem contains nonlinear terms conic constraints. The requested operation cannot be applied to this type of problem.

MSK_RES_ERR_MPS_CONE_OVERLAP (1118)
A variable is speci ed to be a member of several cones.

MSK_RES_ERR_MPS_CONE_REPEAT (1119)
A variable is repeated within the CSECTION

MSK_RES_ERR_MPS_CONE_TYPE (1117)
Invalid cone type speci ed in a CSECTION

MSK_RES_ERR_MPS_DUPLICATE_Q_ELEMENT (1121)
Duplicate elements is spec ed in a Q matrix.

MSK_RES_ERR_MPS_FILE (1100)
An error occurred while reading an MPS le.

MSK_RES_ERR_MPS_INV_BOUND_KEY (1108)
An invalid bound key occurred in an MPS le.

MSK_RES_ERR_MPS_INV_CON_KEY (1107)
An invalid constraint key occurred in an MPS le.

MSK_RES_ERR_MPS_INV_FIELD (1101)
A eld in the MPS le is invalid. Probably it is too wide.

MSK_RES_ERR_MPS_INV_MARKER (1102)
An invalid marker has been speci ed in the MPS le.

MSK_RES_ERR_MPS_INV_SEC_NAME (1109)
An invalid section name occurred in an MPS le.

MSK_RES_ERR_MPS_INV_SEC_ORDER (1115)
The sections in the MPS data le are not in the correct order.

MSK_RES_ERR_MPS_INVALID_OBJ_NAME (1128)
An invalid objective name is speci ed.

MSK_RES_ERR_MPS_INVALID_OBJSENSE (1122)
An invalid objective sense is speci ed.

MSK_RES_ERR_MPS_MUL_CON_NAME (1112)
A constraint name was speci ed multiple times in the ROWS section.

MSK_RES_ERR_MPS_MUL_CSEC (1116)
Multiple CSECTIONS are given the same name.

`MSK_RES_ERR_MPS_MUL_QOBJ (1114)`
The Q term in the objective is specified multiple times in the MPS data file.

`MSK_RES_ERR_MPS_MUL_QSEC (1113)`
Multiple `QSECTION`s are specified for a constraint in the MPS data file.

`MSK_RES_ERR_MPS_NO_OBJECTIVE (1110)`
No objective is defined in an MPS file.

`MSK_RES_ERR_MPS_NON_SYMMETRIC_Q (1120)`
A non symmetric matrice has been speciefied.

`MSK_RES_ERR_MPS_NULL_CON_NAME (1103)`
An empty constraint name is used in an MPS file.

`MSK_RES_ERR_MPS_NULL_VAR_NAME (1104)`
An empty variable name is used in an MPS file.

`MSK_RES_ERR_MPS_SPLITTED_VAR (1111)`
All elements in a column of the A matrix must be specified consecutively. Hence, it is illegal to specify non-zero elements in A for variable 1, then for variable 2 and then variable 1 again.

`MSK_RES_ERR_MPS_TAB_IN_FIELD2 (1125)`
A tab char occurred in field 2.

`MSK_RES_ERR_MPS_TAB_IN_FIELD3 (1126)`
A tab char occurred in field 3.

`MSK_RES_ERR_MPS_TAB_IN_FIELD5 (1127)`
A tab char occurred in field 5.

`MSK_RES_ERR_MPS_UNDEF_CON_NAME (1105)`
An undefined constraint name occurred in an MPS file.

`MSK_RES_ERR_MPS_UNDEF_VAR_NAME (1106)`
An undefined variable name occurred in an MPS file.

`MSK_RES_ERR_MUL_A_ELEMENT (1254)`
An element in A is defined multiple times.

`MSK_RES_ERR_NAME_IS_NULL (1760)`
The name buffer is a NULL pointer.

`MSK_RES_ERR_NAME_MAX_LEN (1750)`
A name is longer than the buffer that is supposed to hold it.

`MSK_RES_ERR_NAN_IN_BLC (1461)`
l^c contains an invalid floating point value, i.e. a `NaN`.

`MSK_RES_ERR_NAN_IN_BLX (1471)`
l^x contains an invalid floating point value, i.e. a `NaN`.

`MSK_RES_ERR_NAN_IN_BUC (1462)`
u^c contains an invalid floating point value, i.e. a `NaN`.

`MSK_RES_ERR_NAN_IN_BUX (1472)`
u^x contains an invalid floating point value, i.e. a `NaN`.

`MSK_RES_ERR_NAN_IN_C (1470)`
c contains an invalid floating point value, i.e. a `NaN`.

`MSK_RES_ERR_NAN_IN_DOUBLE_DATA (1450)`
An invalid floating point value was used in some double data.

`MSK_RES_ERR_NEGATIVE_APPEND (1264)`
Cannot append a negative number.

`MSK_RES_ERR_NEGATIVE_SURPLUS (1263)`
Negative surplus.

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.

16.1.4 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.

16.1.5 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).

16.2 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].

16.2.1 MPS File Structure

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

$$\begin{array}{rcccl} l^c & \leq & Ax + q(x) & \leq & u^c, \\ l^x & \leq & x & \leq & u^x, \\ & & x \in \mathcal{K}, & & \\ & & x_{\mathcal{J}} \text{ integer}, & & \end{array} \tag{16.2}$$

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.
- $q : \mathbb{R}^n \to \mathbb{R}$ 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 elds. 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 Section 16.2.9 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	∞
L	$-\infty$	finite
N	$-\infty$	∞

In the MPS format an objective vector is not specified explicitly, but one of the constraints having the key N will be used as the objective vector c . In general, if multiple N type constraints are specified, then the first will be used as the objective vector c.

COLUMNS

In this section the elements of A are specified using one or more records having the form:

```
[vname1]  [cname1]    [value1]     [cname2]  [value2]
```

where the requirements for each field are as follows:

Field	Starting Position	Max Width	required	Description
[vname1]	5	8	Yes	Variable name
[cname1]	15	8	Yes	Constraint name
[value1]	25	12	Yes	Numerical value
[cname2]	40	8	No	Constraint name
[value2]	50	12	No	Numerical value

Hence, a record specifies one or two elements a_{ij} of A using the principle that [vname1] and [cname1] determines j and i respectively. Please note that [cname1] must be a constraint name specified in the ROWS section. Finally, [value1] denotes the numerical value of a_{ij} . Another optional element is specified by [cname2], and [value2] for the variable specified by [vname1]. Some important comments are:

- All elements belonging to one variable must be grouped together.
- Zero elements of A should not be specified.
- At least one element for each variable should be specified.

RHS (optional)

A record in this section has the format

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

where the requirements for each field are as follows:

Field	Starting Position	Max Width	required	Description
[name]	5	8	Yes	Name of the RHS vector
[cname1]	15	8	Yes	Constraint name
[value1]	25	12	Yes	Numerical value
[cname2]	40	8	No	Constraint name
[value2]	50	12	No	Numerical value

The interpretation of a record is that [name] is the name of the RHS vector to be specified. In general, several vectors can be specified. [cname1] denotes a constraint name previously specified in the ROWS section. Now, assume that this name has been assigned to the i th constraint and v_1 denotes the value specified by [value1], then the interpretation of v_1 is:

Constraint	l_i^c	u_i^c
type		
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

$$\begin{array}{lcl} \text{minimize} & -x_2 + \frac{1}{2}(2x_1^2 - 2x_1x_3 + 0.2x_2^2 + 2x_3^2) & \\ \text{subject to} & x_1 + x_2 + x_3 & \geq \quad 1, \\ & x \geq 0 & \end{array}$$

has the following MPS file representation

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

Regarding the `QSECTION`s please note that:

- Only one QSECTION is allowed for each constraint.
- The 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

$$\begin{array}{lcl} \text{minimize} & -x_2 + \frac{1}{2}(2x_1^2 - 2x_1x_3 + 0.2x_2^2 + 2x_3^2) & \\ \text{subject to} & x_1 + x_2 + x_3 & \geq \quad 1, \\ & x \geq 0 & \end{array}$$

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.

16.2.2 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 $1/2x^T Qx$.

The example

$$\begin{array}{lll} \text{minimize} & x_2 & \\ \text{subject to} & x_1 + x_2 + x_3 & 1; \\ & \frac{1}{2}(2x_1x_3 + 0.2x_2^2 + 2x_3^2) & 10; \\ & x \quad 0 & \end{array}$$

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.

16.2.3 `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`	$-\infty$	∞	No
`FX`	v_1	v_1	No
`LO`	v_1	unchanged	No
`MI`	$-\infty$	unchanged	No
`PL`	unchanged	∞	No
`UP`	unchanged	v_1	No
`BV`	0	1	Yes
`LI`	$\lceil v_1 \rceil$	unchanged	Yes
`UI`	unchanged	$\lfloor v_1 \rfloor$	Yes

`v_1` is the value specified by `[value1]`.

16.2.4 CSECTION (optional)

The purpose of the CSECTION is to specify the constraint

$$x \in \mathcal{K}.$$

in (16.2). 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:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\}. \tag{16.3}$$

- Rotated quadratic cone:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : 2x_1 x_2 \geq \sum_{j=3}^{n^t} x_j^2, \quad x_1, x_2 \geq 0 \right\}. \tag{16.4}$$

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
*234567890123456789012345678901234567890123456789012345678901234567890
CSECTION      konea       0.0          QUAD
x4
x5
x8
CSECTION      koneb       0.0          RQUAD
x7
x3
x1
x0
```

This first CSECTION specifies the cone (16.3) 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 (16.4). 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	≤ 1	Quadratic cone i.e. (16.3).
RQUAD	≤ 2	Rotated quadratic cone i.e. (16.4).

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.

16.2.5 ENDATA

This keyword denotes the end of the MPS file.

16.2.6 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.

16.2.7 General Limitations

- An MPS file should be an ASCII file.

16.2.8 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.

16.2.9 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.

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 16.1.

Listing 16.1: 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

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

This can be formulated in `opf` as shown below.

Listing 16.2: 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:

$$\begin{array}{lcl} \text{minimize} & x_3 + x_4 + x_5 & \\ \text{subject to} & x_0 + x_1 + 2x_2 & = \quad 1, \\ & x_0, x_1, x_2 & \geq \quad 0, \\ & x_3 \geq \sqrt{x_0^2 + x_1^2}, & \\ & 2x_4 x_5 \geq x_2^2. & \end{array}$$

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 16.3.

Listing 16.3: 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:

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

This can be implemented in OPF with the file in Listing 16.4.

Listing 16.4: 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]
```

16.4 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.

16.4.1 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:

$$\begin{array}{lll} \min / \max & g^{obj} & \\ & g_i \in \mathcal{K}_i, & i \in \mathcal{I}, \\ \text{s.t.} & G_i \in \mathcal{K}_i, & i \in \mathcal{I}^{PSD}, \\ & x_j \in \mathcal{K}_j, & j \in \mathcal{J}, \\ & \overline{X}_j \in \mathcal{K}_j, & j \in \mathcal{J}^{PSD}. \end{array} \tag{16.5}$$

- **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

$$\left\{ \begin{pmatrix} p \\ q \\ x \end{pmatrix} \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^{n-2},\ 2pq \geq x^T x,\ p \geq 0,\ q \geq 0 \right\}, \text{ for } n \geq 3.$$

16.4.2 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.

16.4.3 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 16.3. 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 16.3.

- **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 16.3: 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

16.4.4 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 16.3), 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 16.3), 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.

Mixing Linear, Second-order and Semide nite Cones

The conic optimization problem (16.7), has a semide nite cone, a quadratic cone over unordered subindices, and two equality constraints.

$$
\begin{array}{ll}
\text{minimize} & \left\langle \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}, X_1 \right\rangle + x_1 \\
\text{subject to} & \left\langle \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, X_1 \right\rangle + x_1 = 1.0, \\
 & \left\langle \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, X_1 \right\rangle + x_0 + x_2 = 0.5, \\
 & x_1 \geq \sqrt{x_0^2 + x_2^2}, \\
 & X_1 \succeq 0.
\end{array}
\tag{16.7}
$$

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 t under the VAR keyword in this variable permutation. Instead, it takes a scalar constraint $(g_2, g_3, g_4) = (x_1, x_0, x_2) \in Q^3$, with scalar variables constructed as $(x_0, x_1, x_2) \in R^3$. 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.

$$\begin{array}{lll} \text{minimize} & \left\langle \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, X_1 \right\rangle + x_1 + x_2 + 1 & \\ \text{subject to} & \left\langle \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, X_1 \right\rangle - x_1 - x_2 & \geq \quad 0.0\,, \\ & x_1 \begin{bmatrix} 0 & 1 \\ 1 & 3 \end{bmatrix} + x_2 \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \succeq \quad \mathbf{0}\,, \\ & X_1 \succeq \mathbf{0}\,. & \end{array} \tag{16.8}$$

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 (16.9), is de ned for a sequence of objectives such that hotstarting from one to the next might be advantages.

$$\begin{array}{lll} \text{maximize}_k & g_k^{obj} & \\ \text{subject to} & 50x_0 + 31 & 250; \\ & 3x_0 \quad 2x_1 & 4; \\ & x \ 2 \ R_+^2; & \end{array} \tag{16.9}$$

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 16.5.

Listing 16.5: Problem (16.9) 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
```

16.5 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.

16.6 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.

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.

16.7 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.

16.7.1 *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**.

16.7.2 *jsol* format

It stores a problem solution. The jsol format contains all solutions and information items.

16.7.3 A *jtask* example

In Listing 16.6 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 16.6: 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"
        }
    }
}
```


16.8 The Solution File Format

MOSEK provides several solution files depending on the problem type and the optimizer used:

- basis solution le (extension `.bas`) if the problem is optimized using the simplex optimizer or basis identification is performed,
- interior solution le (extension `.sol`) if a problem is optimized using the interior-point optimizer and no basis identification is required,
- integer solution le (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

$$l_i^c \leq \sum_{j=1}^{n} a_{ij} x_j \leq u_i^c, \tag{16.10}$$

  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 16.4 the possible values of the status keys and their interpretation are shown.

Table 16.4: 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 (16.10).)
  - `UPPER LIMIT`: the quantity u_i^c (see (16.10).)
  - `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.

CHAPTER

SEVENTEEN

INTERFACE CHANGES

The section show interface-specific changes to the **MOSEK** MATLAB Toolbox in version 8. See the release notes for general changes and new features of the **MOSEK** Optimization Suite.

17.1 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.

17.2 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*

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_NONCONVEX
- 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_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

17.3 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_CALLBACKCODE_UPDATE_NETWORK_DUAL_SIMPLEX
- MSK_CALLBACKCODE_UPDATE_NETWORK_PRIMAL_SIMPLEX
- MSK_CALLBACKCODE_UPDATE_NONCONVEX
- 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_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_SOL_INT_PROSTA
- MSK_IINFITEM_SOL_INT_SOLSTA
- MSK_IINFITEM_STO_NUM_A_CACHE_FLUSHES
- MSK_IINFITEM_STO_NUM_A_TRANSPOSES

- MSK_MIOMODE_LAZY
- MSK_OPTIMIZERTYPE_CONCURRENT
- MSK_OPTIMIZERTYPE_MIXED_INT_CONIC
- MSK_OPTIMIZERTYPE_NETWORK_PRIMAL_SIMPLEX
- MSK_OPTIMIZERTYPE_NONCONVEX
- MSK_OPTIMIZERTYPE_PRIMAL_DUAL_SIMPLEX

17.4 Response Codes

Added

- MSK_RES_ERR_DUPLICATE_AIJ (1385)
- MSK_RES_ERR_JSON_DATA (1179)
- MSK_RES_ERR_JSON_FORMAT (1178)
- MSK_RES_ERR_JSON_MISSING_DATA (1180)
- MSK_RES_ERR_JSON_NUMBER_OVERFLOW (1177)
- MSK_RES_ERR_JSON_STRING (1176)
- MSK_RES_ERR_JSON_SYNTAX (1175)
- MSK_RES_ERR_LAU_INVALID_LOWER_TRIANGULAR_MATRIX (7002)
- MSK_RES_ERR_LAU_INVALID_SPARSE_SYMMETRIC_MATRIX (7019)
- MSK_RES_ERR_LAU_NOT_POSITIVE_DEFINITE (7001)
- MSK_RES_ERR_MIXED_CONIC_AND_NL (1501)
- MSK_RES_ERR_SERVER_CONNECT (8000)
- MSK_RES_ERR_SERVER_PROTOCOL (8001)
- MSK_RES_ERR_SERVER_STATUS (8002)
- MSK_RES_ERR_SERVER_TOKEN (8003)
- MSK_RES_ERR_SYM_MAT_HUGE (1482)
- MSK_RES_ERR_SYM_MAT_INVALID (1480)
- MSK_RES_ERR_TASK_WRITE (2562)
- MSK_RES_ERR_TOCONIC_CONSTR_NOT_CONIC (7153)
- MSK_RES_ERR_TOCONIC_CONSTR_Q_NOT_PSD (7150)
- MSK_RES_ERR_TOCONIC_CONSTRAINT_FX (7151)
- MSK_RES_ERR_TOCONIC_CONSTRAINT_RA (7152)
- MSK_RES_ERR_TOCONIC_OBJECTIVE_NOT_PSD (7155)
- MSK_RES_WRN_SYM_MAT_LARGE (960)

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_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

[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.

Enumerations

MSKaccmodee, 214
MSK_ACC_CON, 214
MSK_ACC_VAR, 214
MSKbasindtypee, 214
MSK_BI_ALWAYS, 214
MSK_BI_IF_FEASIBLE, 214
MSK_BI_NEVER, 214
MSK_BI_NO_ERROR, 214
MSK_BI_RESERVERED, 214
MSKboundkeye, 214
MSK_BK_FR, 214
MSK_BK_FX, 214
MSK_BK_LO, 214
MSK_BK_RA, 214
MSK_BK_UP, 214
MSKbranchdire, 231
MSK_BRANCH_DIR_DOWN, 232
MSK_BRANCH_DIR_FAR, 232
MSK_BRANCH_DIR_FREE, 231
MSK_BRANCH_DIR_GUIDED, 232
MSK_BRANCH_DIR_NEAR, 232
MSK_BRANCH_DIR_PSEUDOCOST, 232
MSK_BRANCH_DIR_ROOT_LP, 232
MSK_BRANCH_DIR_UP, 232
MSKcallbackcodee, 216
MSK_CALLBACK_BEGIN_BI, 217
MSK_CALLBACK_BEGIN_CONIC, 216
MSK_CALLBACK_BEGIN_DUAL_BI, 217
MSK_CALLBACK_BEGIN_DUAL_SENSITIVITY, 219
MSK_CALLBACK_BEGIN_DUAL_SETUP_BI, 219
MSK_CALLBACK_BEGIN_DUAL_SIMPLEX, 218
MSK_CALLBACK_BEGIN_DUAL_SIMPLEX_BI, 218
MSK_CALLBACK_BEGIN_FULL_CONVEXITY_CHECK, 220
MSK_CALLBACK_BEGIN_INFEAS_ANA, 219
MSK_CALLBACK_BEGIN_INTPNT, 216
MSK_CALLBACK_BEGIN_LICENSE_WAIT, 220
MSK_CALLBACK_BEGIN_MIO, 218
MSK_CALLBACK_BEGIN_OPTIMIZER, 216
MSK_CALLBACK_BEGIN_PRESOLVE, 216
MSK_CALLBACK_BEGIN_PRIMAL_BI, 217
MSK_CALLBACK_BEGIN_PRIMAL_DUAL_SIMPLEX, 219
MSK_CALLBACK_BEGIN_PRIMAL_DUAL_SIMPLEX_BI, 218
MSK_CALLBACK_BEGIN_PRIMAL_REPAIR, 220
MSK_CALLBACK_BEGIN_PRIMAL_SENSITIVITY, 219
MSK_CALLBACK_BEGIN_PRIMAL_SETUP_BI, 219
MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX, 218
MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX_BI, 217
MSK_CALLBACK_BEGIN_QCQO_REFORMULATE, 220
MSK_CALLBACK_BEGIN_READ, 220
MSK_CALLBACK_BEGIN_ROOT_CUTGEN, 216
MSK_CALLBACK_BEGIN_SIMPLEX, 218
MSK_CALLBACK_BEGIN_SIMPLEX_BI, 217
MSK_CALLBACK_BEGIN_TO_CONIC, 220
MSK_CALLBACK_BEGIN_WRITE, 220
MSK_CALLBACK_CONIC, 216
MSK_CALLBACK_DUAL_SIMPLEX, 217
MSK_CALLBACK_END_BI, 217
MSK_CALLBACK_END_CONIC, 217
MSK_CALLBACK_END_DUAL_BI, 217
MSK_CALLBACK_END_DUAL_SENSITIVITY, 219
MSK_CALLBACK_END_DUAL_SETUP_BI, 219
MSK_CALLBACK_END_DUAL_SIMPLEX, 218
MSK_CALLBACK_END_DUAL_SIMPLEX_BI, 218
MSK_CALLBACK_END_FULL_CONVEXITY_CHECK, 220
MSK_CALLBACK_END_INFEAS_ANA, 219
MSK_CALLBACK_END_INTPNT, 216
MSK_CALLBACK_END_LICENSE_WAIT, 220
MSK_CALLBACK_END_MIO, 218
MSK_CALLBACK_END_OPTIMIZER, 216
MSK_CALLBACK_END_PRESOLVE, 216
MSK_CALLBACK_END_PRIMAL_BI, 217
MSK_CALLBACK_END_PRIMAL_DUAL_SIMPLEX, 219
MSK_CALLBACK_END_PRIMAL_DUAL_SIMPLEX_BI, 218
MSK_CALLBACK_END_PRIMAL_REPAIR, 220
MSK_CALLBACK_END_PRIMAL_SENSITIVITY, 219
MSK_CALLBACK_END_PRIMAL_SETUP_BI, 219
MSK_CALLBACK_END_PRIMAL_SIMPLEX, 219
MSK_CALLBACK_END_PRIMAL_SIMPLEX_BI, 218
MSK_CALLBACK_END_QCQO_REFORMULATE, 220
MSK_CALLBACK_END_READ, 220
MSK_CALLBACK_END_ROOT_CUTGEN, 216
MSK_CALLBACK_END_SIMPLEX, 219
MSK_CALLBACK_END_SIMPLEX_BI, 218
MSK_CALLBACK_END_TO_CONIC, 220
MSK_CALLBACK_END_WRITE, 220
MSK_CALLBACK_IM_BI, 217
MSK_CALLBACK_IM_CONIC, 216
MSK_CALLBACK_IM_DUAL_BI, 217
MSK_CALLBACK_IM_DUAL_SENSIVITY, 219

MSK_CALLBACK_IM_DUAL_SIMPLEX, 218
MSK_CALLBACK_IM_FULL_CONVEXITY_CHECK, 220
MSK_CALLBACK_IM_INTPNT, 216
MSK_CALLBACK_IM_LICENSE_WAIT, 220
MSK_CALLBACK_IM_LU, 220
MSK_CALLBACK_IM_MIO, 218
MSK_CALLBACK_IM_MIO_DUAL_SIMPLEX, 219
MSK_CALLBACK_IM_MIO_INTPNT, 219
MSK_CALLBACK_IM_MIO_PRIMAL_SIMPLEX, 219
MSK_CALLBACK_IM_ORDER, 220
MSK_CALLBACK_IM_PRESOLVE, 216
MSK_CALLBACK_IM_PRIMAL_BI, 217
MSK_CALLBACK_IM_PRIMAL_DUAL_SIMPLEX, 219
MSK_CALLBACK_IM_PRIMAL_SENSIVITY, 219
MSK_CALLBACK_IM_PRIMAL_SIMPLEX, 218
MSK_CALLBACK_IM_QO_REFORMULATE, 220
MSK_CALLBACK_IM_READ, 220
MSK_CALLBACK_IM_ROOT_CUTGEN, 216
MSK_CALLBACK_IM_SIMPLEX, 220
MSK_CALLBACK_IM_SIMPLEX_BI, 217
MSK_CALLBACK_INTPNT, 216
MSK_CALLBACK_NEW_INT_MIO, 218
MSK_CALLBACK_PRIMAL_SIMPLEX, 217
MSK_CALLBACK_READ_OPF, 221
MSK_CALLBACK_READ_OPF_SECTION, 220
MSK_CALLBACK_SOLVING_REMOTE, 221
MSK_CALLBACK_UPDATE_DUAL_BI, 217
MSK_CALLBACK_UPDATE_DUAL_SIMPLEX, 218
MSK_CALLBACK_UPDATE_DUAL_SIMPLEX_BI, 218
MSK_CALLBACK_UPDATE_PRESOLVE, 216
MSK_CALLBACK_UPDATE_PRIMAL_BI, 217
MSK_CALLBACK_UPDATE_PRIMAL_DUAL_SIMPLEX, 219
MSK_CALLBACK_UPDATE_PRIMAL_DUAL_SIMPLEX_BI, 218
MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX, 219
MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX_BI, 217
MSK_CALLBACK_WRITE_OPF, 221
MSKcheckconvexitytypee, 221
MSK_CHECK_CONVEXITY_FULL, 221
MSK_CHECK_CONVEXITY_NONE, 221
MSK_CHECK_CONVEXITY_SIMPLE, 221
MSKcompresstypee, 221
MSK_COMPRESS_FREE, 221
MSK_COMPRESS_GZIP, 221
MSK_COMPRESS_NONE, 221
MSKconetypee, 221
MSK_CT_QUAD, 221
MSK_CT_RQUAD, 221
MSKdataformate, 221
MSK_DATA_FORMAT_CB, 222
MSK_DATA_FORMAT_EXTENSION, 221
MSK_DATA_FORMAT_FREE_MPS, 222
MSK_DATA_FORMAT_JSON_TASK, 222
MSK_DATA_FORMAT_LP, 222
MSK_DATA_FORMAT_MPS, 221
MSK_DATA_FORMAT_OP, 222
MSK_DATA_FORMAT_TASK, 222
MSK_DATA_FORMAT_XML, 222
MSKdinfiteme, 222
MSK_DINF_BI_CLEAN_DUAL_TIME, 222
MSK_DINF_BI_CLEAN_PRIMAL_DUAL_TIME, 222
MSK_DINF_BI_CLEAN_PRIMAL_TIME, 222
MSK_DINF_BI_CLEAN_TIME, 222
MSK_DINF_BI_DUAL_TIME, 222
MSK_DINF_BI_PRIMAL_TIME, 222
MSK_DINF_BI_TIME, 222
MSK_DINF_INTPNT_DUAL_FEAS, 222
MSK_DINF_INTPNT_DUAL_OBJ, 222
MSK_DINF_INTPNT_FACTOR_NUM_FLOPS, 226
MSK_DINF_INTPNT_OPT_STATUS, 223
MSK_DINF_INTPNT_ORDER_TIME, 222
MSK_DINF_INTPNT_PRIMAL_FEAS, 222
MSK_DINF_INTPNT_PRIMAL_OBJ, 222
MSK_DINF_INTPNT_TIME, 222
MSK_DINF_MIO_CLIQUE_SEPARATION_TIME, 224
MSK_DINF_MIO_CMIR_SEPARATION_TIME, 224
MSK_DINF_MIO_CONSTRUCT_SOLUTION_OBJ, 223
MSK_DINF_MIO_DUAL_BOUND_AFTER_PRESOLVE, 224
MSK_DINF_MIO_GMI_SEPARATION_TIME, 224
MSK_DINF_MIO_HEURISTIC_TIME, 223
MSK_DINF_MIO_IMPLIED_BOUND_TIME, 224
MSK_DINF_MIO_KNAPSACK_COVER_SEPARATION_TIME, 224
MSK_DINF_MIO_OBJ_ABS_GAP, 224
MSK_DINF_MIO_OBJ_BOUND, 223
MSK_DINF_MIO_OBJ_INT, 223
MSK_DINF_MIO_OBJ_REL_GAP, 223
MSK_DINF_MIO_OPTIMIZER_TIME, 223
MSK_DINF_MIO_PROBING_TIME, 224
MSK_DINF_MIO_ROOT_CUTGEN_TIME, 224
MSK_DINF_MIO_ROOT_OPTIMIZER_TIME, 223
MSK_DINF_MIO_ROOT_PRESOLVE_TIME, 223
MSK_DINF_MIO_TIME, 223
MSK_DINF_MIO_USER_OBJ_CUT, 224
MSK_DINF_OPTIMIZER_TIME, 224
MSK_DINF_PRESOLVE_ELI_TIME, 224
MSK_DINF_PRESOLVE_LINDEP_TIME, 224
MSK_DINF_PRESOLVE_TIME, 224
MSK_DINF_PRIMAL_REPAIR_PENALTY_OBJ, 226
MSK_DINF_QCQO_REFORMULATE_MAX_PERTURBATION, 226
MSK_DINF_QCQO_REFORMULATE_TIME, 226
MSK_DINF_QCQO_REFORMULATE_WORST_CHOLESKY_COLUMN_SCALING, 226
MSK_DINF_QCQO_REFORMULATE_WORST_CHOLESKY_DIAG_SCALING, 226
MSK_DINF_RD_TIME, 224
MSK_DINF_SIM_DUAL_TIME, 223
MSK_DINF_SIM_FEAS, 223
MSK_DINF_SIM_OBJ, 223
MSK_DINF_SIM_PRIMAL_DUAL_TIME, 223
MSK_DINF_SIM_PRIMAL_TIME, 223
MSK_DINF_SIM_TIME, 223
MSK_DINF_SOL_BAS_DUAL_OBJ, 225
MSK_DINF_SOL_BAS_DVIOLCON, 225

MSK_DINF_SOL_BAS_DVIOLVAR, 225
MSK_DINF_SOL_BAS_NRM_BARX, 225
MSK_DINF_SOL_BAS_NRM_SLC, 226
MSK_DINF_SOL_BAS_NRM_SLX, 226
MSK_DINF_SOL_BAS_NRM_SUC, 226
MSK_DINF_SOL_BAS_NRM_SUX, 226
MSK_DINF_SOL_BAS_NRM_XC, 225
MSK_DINF_SOL_BAS_NRM_XX, 225
MSK_DINF_SOL_BAS_NRM_Y, 226
MSK_DINF_SOL_BAS_PRIMAL_OBJ, 225
MSK_DINF_SOL_BAS_PVIOLCON, 225
MSK_DINF_SOL_BAS_PVIOLVAR, 225
MSK_DINF_SOL_ITG_NRM_BARX, 226
MSK_DINF_SOL_ITG_NRM_XC, 226
MSK_DINF_SOL_ITG_NRM_XX, 226
MSK_DINF_SOL_ITG_PRIMAL_OBJ, 226
MSK_DINF_SOL_ITG_PVIOLBARVAR, 226
MSK_DINF_SOL_ITG_PVIOLCON, 226
MSK_DINF_SOL_ITG_PVIOLCONES, 226
MSK_DINF_SOL_ITG_PVIOLITG, 226
MSK_DINF_SOL_ITG_PVIOLVAR, 226
MSK_DINF_SOL_ITR_DUAL_OBJ, 224
MSK_DINF_SOL_ITR_DVIOLBARVAR, 225
MSK_DINF_SOL_ITR_DVIOLCON, 225
MSK_DINF_SOL_ITR_DVIOLCONES, 225
MSK_DINF_SOL_ITR_DVIOLVAR, 225
MSK_DINF_SOL_ITR_NRM_BARS, 225
MSK_DINF_SOL_ITR_NRM_BARX, 225
MSK_DINF_SOL_ITR_NRM_SLC, 225
MSK_DINF_SOL_ITR_NRM_SLX, 225
MSK_DINF_SOL_ITR_NRM_SNX, 225
MSK_DINF_SOL_ITR_NRM_SUC, 225
MSK_DINF_SOL_ITR_NRM_SUX, 225
MSK_DINF_SOL_ITR_NRM_XC, 225
MSK_DINF_SOL_ITR_NRM_XX, 225
MSK_DINF_SOL_ITR_NRM_Y, 225
MSK_DINF_SOL_ITR_PRIMAL_OBJ, 224
MSK_DINF_SOL_ITR_PVIOLBARVAR, 224
MSK_DINF_SOL_ITR_PVIOLCON, 224
MSK_DINF_SOL_ITR_PVIOLCONES, 224
MSK_DINF_SOL_ITR_PVIOLVAR, 224
MSK_DINF_TO_CONIC_TIME, 223
MSKfeaturee, 226
MSK_FEATURE_PTON, 226
MSK_FEATURE_PTS, 226
MSKiinfiteme, 227
MSK_IINF_ANA_PRO_NUM_CON, 227
MSK_IINF_ANA_PRO_NUM_CON_EQ, 227
MSK_IINF_ANA_PRO_NUM_CON_FR, 228
MSK_IINF_ANA_PRO_NUM_CON_LO, 227
MSK_IINF_ANA_PRO_NUM_CON_RA, 227
MSK_IINF_ANA_PRO_NUM_CON_UP, 227
MSK_IINF_ANA_PRO_NUM_VAR, 228
MSK_IINF_ANA_PRO_NUM_VAR_BIN, 228
MSK_IINF_ANA_PRO_NUM_VAR_CONT, 228
MSK_IINF_ANA_PRO_NUM_VAR_EQ, 228
MSK_IINF_ANA_PRO_NUM_VAR_FR, 228
MSK_IINF_ANA_PRO_NUM_VAR_INT, 228
MSK_IINF_ANA_PRO_NUM_VAR_LO, 228
MSK_IINF_ANA_PRO_NUM_VAR_RA, 228
MSK_IINF_ANA_PRO_NUM_VAR_UP, 228
MSK_IINF_INTPNT_FACTOR_DIM_DENSE, 228
MSK_IINF_INTPNT_ITER, 228
MSK_IINF_INTPNT_NUM_THREADS, 230
MSK_IINF_INTPNT_SOLVE_DUAL, 228
MSK_IINF_MIO_ABSGAP_SATISFIED, 229
MSK_IINF_MIO_CLIQUE_TABLE_SIZE, 229
MSK_IINF_MIO_CONSTRUCT_NUM_ROUNDINGS, 229
MSK_IINF_MIO_CONSTRUCT_SOLUTION, 229
MSK_IINF_MIO_INITIAL_SOLUTION, 229
MSK_IINF_MIO_NEAR_ABSGAP_SATISFIED, 229
MSK_IINF_MIO_NEAR_RELGAP_SATISFIED, 229
MSK_IINF_MIO_NODE_DEPTH, 228
MSK_IINF_MIO_NUM_ACTIVE_NODES, 229
MSK_IINF_MIO_NUM_BRANCH, 229
MSK_IINF_MIO_NUM_CLIQUE_CUTS, 229
MSK_IINF_MIO_NUM_CMIR_CUTS, 229
MSK_IINF_MIO_NUM_GOMORY_CUTS, 229
MSK_IINF_MIO_NUM_IMPLIED_BOUND_CUTS, 229
MSK_IINF_MIO_NUM_INT_SOLUTIONS, 229
MSK_IINF_MIO_NUM_KNAPSACK_COVER_CUTS, 229
MSK_IINF_MIO_NUM_RELAX, 229
MSK_IINF_MIO_NUM_REPEATED_PRESOLVE, 229
MSK_IINF_MIO_NUMCON, 228
MSK_IINF_MIO_NUMINT, 228
MSK_IINF_MIO_NUMVAR, 228
MSK_IINF_MIO_OBJ_BOUND_DEFINED, 229
MSK_IINF_MIO_PRESOLVED_NUMBIN, 228
MSK_IINF_MIO_PRESOLVED_NUMCON, 228
MSK_IINF_MIO_PRESOLVED_NUMCONT, 228
MSK_IINF_MIO_PRESOLVED_NUMINT, 228
MSK_IINF_MIO_PRESOLVED_NUMVAR, 228
MSK_IINF_MIO_RELGAP_SATISFIED, 229
MSK_IINF_MIO_TOTAL_NUM_CUTS, 229
MSK_IINF_MIO_USER_OBJ_CUT, 229
MSK_IINF_OPT_NUMCON, 231
MSK_IINF_OPT_NUMVAR, 231
MSK_IINF_OPTIMIZE_RESPONSE, 228
MSK_IINF_RD_NUMBARVAR, 230
MSK_IINF_RD_NUMCON, 230
MSK_IINF_RD_NUMCONE, 231
MSK_IINF_RD_NUMINTVAR, 230
MSK_IINF_RD_NUMQ, 230
MSK_IINF_RD_NUMVAR, 230
MSK_IINF_RD_PROTYPE, 229
MSK_IINF_SIM_DUAL_DEG_ITER, 230
MSK_IINF_SIM_DUAL_HOTSTART, 230
MSK_IINF_SIM_DUAL_HOTSTART_LU, 230
MSK_IINF_SIM_DUAL_INF_ITER, 230
MSK_IINF_SIM_DUAL_ITER, 230
MSK_IINF_SIM_NUMCON, 231
MSK_IINF_SIM_NUMVAR, 231
MSK_IINF_SIM_PRIMAL_DEG_ITER, 230
MSK_IINF_SIM_PRIMAL_DUAL_DEG_ITER, 230
MSK_IINF_SIM_PRIMAL_DUAL_HOTSTART, 230
MSK_IINF_SIM_PRIMAL_DUAL_HOTSTART_LU, 230

MSK_IINF_SIM_PRIMAL_DUAL_INF_ITER, 230
MSK_IINF_SIM_PRIMAL_DUAL_ITER, 230
MSK_IINF_SIM_PRIMAL_HOTSTART, 230
MSK_IINF_SIM_PRIMAL_HOTSTART_LU, 230
MSK_IINF_SIM_PRIMAL_INF_ITER, 230
MSK_IINF_SIM_PRIMAL_ITER, 230
MSK_IINF_SIM_SOLVE_DUAL, 231
MSK_IINF_SOL_BAS_PROSTA, 231
MSK_IINF_SOL_BAS_SOLSTA, 231
MSK_IINF_SOL_ITG_PROSTA, 231
MSK_IINF_SOL_ITG_SOLSTA, 231
MSK_IINF_SOL_ITR_PROSTA, 231
MSK_IINF_SOL_ITR_SOLSTA, 231
MSK_IINF_STO_NUM_A_REALLOC, 231
MSKinftypee, 231
MSK_INF_DOU_TYPE, 231
MSK_INF_INT_TYPE, 231
MSK_INF_LINT_TYPE, 231
MSKintpnthotstarte, 215
MSK_INTPNT_HOTSTART_DUAL, 216
MSK_INTPNT_HOTSTART_NONE, 216
MSK_INTPNT_HOTSTART_PRIMAL, 216
MSK_INTPNT_HOTSTART_PRIMAL_DUAL, 216
MSKiomodee, 231
MSK_IOMODE_READ, 231
MSK_IOMODE_READWRITE, 231
MSK_IOMODE_WRITE, 231
MSKlanguagee, 213
MSK_LANG_DAN, 214
MSK_LANG_ENG, 213
MSKliinfiteme, 227
MSK_LIINF_BI_CLEAN_DUAL_DEG_ITER, 227
MSK_LIINF_BI_CLEAN_DUAL_ITER, 227
MSK_LIINF_BI_CLEAN_PRIMAL_DEG_ITER, 227
MSK_LIINF_BI_CLEAN_PRIMAL_DUAL_DEG_ITER, 227
MSK_LIINF_BI_CLEAN_PRIMAL_DUAL_ITER, 227
MSK_LIINF_BI_CLEAN_PRIMAL_DUAL_SUB_ITER, 227
MSK_LIINF_BI_CLEAN_PRIMAL_ITER, 227
MSK_LIINF_BI_DUAL_ITER, 227
MSK_LIINF_BI_PRIMAL_ITER, 227
MSK_LIINF_INTPNT_FACTOR_NUM_NZ, 227
MSK_LIINF_MIO_INTPNT_ITER, 227
MSK_LIINF_MIO_PRESOLVED_ANZ, 227
MSK_LIINF_MIO_SIM_MAXITER_SETBACKS, 227
MSK_LIINF_MIO_SIMPLEX_ITER, 227
MSK_LIINF_RD_NUMANZ, 227
MSK_LIINF_RD_NUMQNZ, 227
MSKmarke, 214
MSK_MARK_LO, 214
MSK_MARK_UP, 214
MSKmiocontsoltypee, 232
MSK_MIO_CONT_SOL_ITG, 232
MSK_MIO_CONT_SOL_ITG_REL, 232
MSK_MIO_CONT_SOL_NONE, 232
MSK_MIO_CONT_SOL_ROOT, 232
MSKmiomodee, 232
MSK_MIO_MODE_IGNORED, 232
MSK_MIO_MODE_SATISFIED, 232
MSKmionodeseltypee, 232
MSK_MIO_NODE_SELECTION_BEST, 232
MSK_MIO_NODE_SELECTION_FIRST, 232
MSK_MIO_NODE_SELECTION_FREE, 232
MSK_MIO_NODE_SELECTION_HYBRID, 232
MSK_MIO_NODE_SELECTION_PSEUDO, 233
MSK_MIO_NODE_SELECTION_WORST, 232
MSKmpsformate, 233
MSK_MPS_FORMAT_CPLEX, 233
MSK_MPS_FORMAT_FREE, 233
MSK_MPS_FORMAT_RELAXED, 233
MSK_MPS_FORMAT_STRICT, 233
MSKmsgkeye, 233
MSK_MSG_MPS_SELECTED, 233
MSK_MSG_READING_FILE, 233
MSK_MSG_WRITING_FILE, 233
MSKnametypee, 221
MSK_NAME_TYPE_GEN, 221
MSK_NAME_TYPE_LP, 221
MSK_NAME_TYPE_MPS, 221
MSKobjsensee, 233
MSK_OBJECTIVE_SENSE_MAXIMIZE, 233
MSK_OBJECTIVE_SENSE_MINIMIZE, 233
MSKonoffkeye, 233
MSK_OFF, 233
MSK_ON, 233
MSKoptimizertypee, 233
MSK_OPTIMIZER_CONIC, 233
MSK_OPTIMIZER_DUAL_SIMPLEX, 233
MSK_OPTIMIZER_FREE, 233
MSK_OPTIMIZER_FREE_SIMPLEX, 233
MSK_OPTIMIZER_INTPNT, 233
MSK_OPTIMIZER_MIXED_INT, 233
MSK_OPTIMIZER_PRIMAL_SIMPLEX, 233
MSKorderingtypee, 234
MSK_ORDER_METHOD_APPMINLOC, 234
MSK_ORDER_METHOD_EXPERIMENTAL, 234
MSK_ORDER_METHOD_FORCE_GRAPHPAR, 234
MSK_ORDER_METHOD_FREE, 234
MSK_ORDER_METHOD_NONE, 234
MSK_ORDER_METHOD_TRY_GRAPHPAR, 234
MSKparametertypee, 234
MSK_PAR_DOU_TYPE, 234
MSK_PAR_INT_TYPE, 234
MSK_PAR_INVALID_TYPE, 234
MSK_PAR_STR_TYPE, 234
MSKpresolvemodee, 234
MSK_PRESOLVE_MODE_FREE, 234
MSK_PRESOLVE_MODE_OFF, 234
MSK_PRESOLVE_MODE_ON, 234
MSKproblemiteme, 234
MSK_PI_CON, 234
MSK_PI_CONE, 234
MSK_PI_VAR, 234
MSKproblemtypee, 234
MSK_PROBTYPE_CONIC, 235

MSK_PROBTYPE_GECO, 235
MSK_PROBTYPE_LO, 234
MSK_PROBTYPE_MIXED, 235
MSK_PROBTYPE_QCQO, 235
MSK_PROBTYPE_QO, 234
MSKprostae, 235
MSK_PRO_STA_DUAL_FEAS, 235
MSK_PRO_STA_DUAL_INFEAS, 235
MSK_PRO_STA_ILL_POSED, 235
MSK_PRO_STA_NEAR_DUAL_FEAS, 235
MSK_PRO_STA_NEAR_PRIM_AND_DUAL_FEAS, 235
MSK_PRO_STA_NEAR_PRIM_FEAS, 235
MSK_PRO_STA_PRIM_AND_DUAL_FEAS, 235
MSK_PRO_STA_PRIM_AND_DUAL_INFEAS, 235
MSK_PRO_STA_PRIM_FEAS, 235
MSK_PRO_STA_PRIM_INFEAS, 235
MSK_PRO_STA_PRIM_INFEAS_OR_UNBOUNDED, 235
MSK_PRO_STA_UNKNOWN, 235
MSKrescodetypee, 235
MSK_RESPONSE_ERR, 236
MSK_RESPONSE_OK, 235
MSK_RESPONSE_TRM, 236
MSK_RESPONSE_UNK, 236
MSK_RESPONSE_WRN, 236
MSKscalingmethode, 236
MSK_SCALING_METHOD_FREE, 236
MSK_SCALING_METHOD_POW2, 236
MSKscalingtypee, 236
MSK_SCALING_AGGRESSIVE, 236
MSK_SCALING_FREE, 236
MSK_SCALING_MODERATE, 236
MSK_SCALING_NONE, 236
MSKsensitivitytypee, 236
MSK_SENSITIVITY_TYPE_BASIS, 236
MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION, 236
MSKsimdegene, 214
MSK_SIM_DEGEN_AGGRESSIVE, 215
MSK_SIM_DEGEN_FREE, 214
MSK_SIM_DEGEN_MINIMUM, 215
MSK_SIM_DEGEN_MODERATE, 215
MSK_SIM_DEGEN_NONE, 214
MSKsimdupvece, 215
MSK_SIM_EXPLOIT_DUPVEC_FREE, 215
MSK_SIM_EXPLOIT_DUPVEC_OFF, 215
MSK_SIM_EXPLOIT_DUPVEC_ON, 215
MSKsimhotstarte, 215
MSK_SIM_HOTSTART_FREE, 215
MSK_SIM_HOTSTART_NONE, 215
MSK_SIM_HOTSTART_STATUS_KEYS, 215
MSKsimreforme, 215
MSK_SIM_REFORMULATION_AGGRESSIVE, 215
MSK_SIM_REFORMULATION_FREE, 215
MSK_SIM_REFORMULATION_OFF, 215
MSK_SIM_REFORMULATION_ON, 215
MSKsimseltypee, 236
MSK_SIM_SELECTION_ASE, 236
MSK_SIM_SELECTION_DEVEX, 236
MSK_SIM_SELECTION_FREE, 236
MSK_SIM_SELECTION_FULL, 236
MSK_SIM_SELECTION_PARTIAL, 236
MSK_SIM_SELECTION_SE, 236
MSKsoliteme, 237
MSK_SOL_ITEM_SLC, 237
MSK_SOL_ITEM_SLX, 237
MSK_SOL_ITEM_SNX, 237
MSK_SOL_ITEM_SUC, 237
MSK_SOL_ITEM_SUX, 237
MSK_SOL_ITEM_XC, 237
MSK_SOL_ITEM_XX, 237
MSK_SOL_ITEM_Y, 237
MSKsolstae, 237
MSK_SOL_STA_DUAL_FEAS, 237
MSK_SOL_STA_DUAL_ILLPOSED_CER, 238
MSK_SOL_STA_DUAL_INFEAS_CER, 237
MSK_SOL_STA_INTEGER_OPTIMAL, 238
MSK_SOL_STA_NEAR_DUAL_FEAS, 237
MSK_SOL_STA_NEAR_DUAL_INFEAS_CER, 237
MSK_SOL_STA_NEAR_INTEGER_OPTIMAL, 238
MSK_SOL_STA_NEAR_OPTIMAL, 237
MSK_SOL_STA_NEAR_PRIM_AND_DUAL_FEAS, 237
MSK_SOL_STA_NEAR_PRIM_FEAS, 237
MSK_SOL_STA_NEAR_PRIM_INFEAS_CER, 237
MSK_SOL_STA_OPTIMAL, 237
MSK_SOL_STA_PRIM_AND_DUAL_FEAS, 237
MSK_SOL_STA_PRIM_FEAS, 237
MSK_SOL_STA_PRIM_ILLPOSED_CER, 238
MSK_SOL_STA_PRIM_INFEAS_CER, 237
MSK_SOL_STA_UNKNOWN, 237
MSKsoltypee, 238
MSK_SOL_BAS, 238
MSK_SOL_ITG, 238
MSK_SOL_ITR, 238
MSKsolveforme, 238
MSK_SOLVE_DUAL, 238
MSK_SOLVE_FREE, 238
MSK_SOLVE_PRIMAL, 238
MSKstakeye, 238
MSK_SK_BAS, 238
MSK_SK_FIX, 238
MSK_SK_INF, 238
MSK_SK_LOW, 238
MSK_SK_SUPBAS, 238
MSK_SK_UNK, 238
MSK_SK_UPR, 238
MSKstartpointtypee, 238
MSK_STARTING_POINT_CONSTANT, 239
MSK_STARTING_POINT_FREE, 238
MSK_STARTING_POINT_GUESS, 238
MSK_STARTING_POINT_SATISFY_BOUNDS, 239
MSKstreamtypee, 239
MSK_STREAM_ERR, 239
MSK_STREAM_LOG, 239
MSK_STREAM_MSG, 239
MSK_STREAM_WRN, 239
MSKsymmattypee, 221
MSK_SYMMAT_TYPE_SPARSE, 221

MSKtransposee, 215
MSK_TRANSPOSE_NO, 215
MSK_TRANSPOSE_YES, 215
MSKuploe, 215
MSK_UPLO_LO, 215
MSK_UPLO_UP, 215
MSKvaluee, 239
MSK_LICENSE_BUFFER_LENGTH, 239
MSK_MAX_STR_LEN, 239
MSKvariabletypee, 239
MSK_VAR_TYPE_CONT, 239
MSK_VAR_TYPE_INT, 239
MSKxmlwriteroutputtypee, 235
MSK_WRITE_XML_MODE_COL, 235
MSK_WRITE_XML_MODE_ROW, 235

Functions

intlinprog (intlinprog) , 130
linprog (linprog) , 131
lsqlin (lsqlin) , 132
lsqnonneg (lsqnonneg), 134
mosekopt (mosekopt), 126
mskenopt (mskenopt), 128
mskgpopt (mskgpopt), 128
mskgpread (mskgpread), 129
mskgpwri (mskgpwri), 129
msklpopt (msklpopt), 127
mskoptimget (mskoptimget), 130
mskoptimset (mskoptimset), 130
mskqpopt (mskqpopt), 127
mskscopt (mskscopt), 128
quadprog (quadprog), 134

Parameters

Double params, 145
MSK_DPAR_ANA_SOL_INFEAS_TOL, 145
MSK_DPAR_BASIS_REL_TOL_S, 145
MSK_DPAR_BASIS_TOL_S, 145
MSK_DPAR_BASIS_TOL_X, 145
MSK_DPAR_CHECK_CONVEXITY_REL_TOL, 145
MSK_DPAR_DATA_SYM_MAT_TOL, 146
MSK_DPAR_DATA_SYM_MAT_TOL_HUGE, 146
MSK_DPAR_DATA_SYM_MAT_TOL_LARGE, 146
MSK_DPAR_DATA_TOL_AIJ, 146
MSK_DPAR_DATA_TOL_AIJ_HUGE, 146
MSK_DPAR_DATA_TOL_AIJ_LARGE, 146
MSK_DPAR_DATA_TOL_BOUND_INF, 147
MSK_DPAR_DATA_TOL_BOUND_WRN, 147
MSK_DPAR_DATA_TOL_C_HUGE, 147
MSK_DPAR_DATA_TOL_CJ_LARGE, 147
MSK_DPAR_DATA_TOL_QIJ, 147
MSK_DPAR_DATA_TOL_X, 147
MSK_DPAR_INTPNT_CO_TOL_DFEAS, 147
MSK_DPAR_INTPNT_CO_TOL_INFEAS, 147
MSK_DPAR_INTPNT_CO_TOL_MU_RED, 148
MSK_DPAR_INTPNT_CO_TOL_NEAR_REL, 148
MSK_DPAR_INTPNT_CO_TOL_PFEAS, 148
MSK_DPAR_INTPNT_CO_TOL_REL_GAP, 148
MSK_DPAR_INTPNT_NL_MERIT_BAL, 148
MSK_DPAR_INTPNT_NL_TOL_DFEAS, 148
MSK_DPAR_INTPNT_NL_TOL_MU_RED, 148
MSK_DPAR_INTPNT_NL_TOL_NEAR_REL, 149
MSK_DPAR_INTPNT_NL_TOL_PFEAS, 149
MSK_DPAR_INTPNT_NL_TOL_REL_GAP, 149
MSK_DPAR_INTPNT_NL_TOL_REL_STEP, 149
MSK_DPAR_INTPNT_QO_TOL_DFEAS, 149
MSK_DPAR_INTPNT_QO_TOL_INFEAS, 149
MSK_DPAR_INTPNT_QO_TOL_MU_RED, 149
MSK_DPAR_INTPNT_QO_TOL_NEAR_REL, 149
MSK_DPAR_INTPNT_QO_TOL_PFEAS, 150
MSK_DPAR_INTPNT_QO_TOL_REL_GAP, 150
MSK_DPAR_INTPNT_TOL_DFEAS, 150
MSK_DPAR_INTPNT_TOL_DSAFE, 150
MSK_DPAR_INTPNT_TOL_INFEAS, 150
MSK_DPAR_INTPNT_TOL_MU_RED, 150
MSK_DPAR_INTPNT_TOL_PATH, 150
MSK_DPAR_INTPNT_TOL_PFEAS, 151
MSK_DPAR_INTPNT_TOL_PSAFE, 151
MSK_DPAR_INTPNT_TOL_REL_GAP, 151
MSK_DPAR_INTPNT_TOL_REL_STEP, 151
MSK_DPAR_INTPNT_TOL_STEP_SIZE, 151
MSK_DPAR_LOWER_OBJ_CUT, 151
MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH, 151
MSK_DPAR_MIO_DISABLE_TERM_TIME, 152
MSK_DPAR_MIO_MAX_TIME, 152
MSK_DPAR_MIO_NEAR_TOL_ABS_GAP, 152
MSK_DPAR_MIO_NEAR_TOL_REL_GAP, 152
MSK_DPAR_MIO_REL_GAP_CONST, 152
MSK_DPAR_MIO_TOL_ABS_GAP, 152
MSK_DPAR_MIO_TOL_ABS_RELAX_INT, 153
MSK_DPAR_MIO_TOL_FEAS, 153
MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT, 153
MSK_DPAR_MIO_TOL_REL_GAP, 153
MSK_DPAR_OPTIMIZER_MAX_TIME, 153
MSK_DPAR_PRESOLVE_TOL_ABS_LINDEP, 153
MSK_DPAR_PRESOLVE_TOL_AIJ, 153
MSK_DPAR_PRESOLVE_TOL_REL_LINDEP, 153
MSK_DPAR_PRESOLVE_TOL_S, 154
MSK_DPAR_PRESOLVE_TOL_X, 154
MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL, 154
MSK_DPAR_SEMIDEFINITE_TOL_APPROX, 154
MSK_DPAR_SIM_LU_TOL_REL_PIV, 154
MSK_DPAR_SIMPLEX_ABS_TOL_PIV, 154
MSK_DPAR_UPPER_OBJ_CUT, 154
MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH, 154
Integer params, 155
MSK_IPAR_ANA_SOL_BASIS, 155
MSK_IPAR_ANA_SOL_PRINT_VIOLATED, 155
MSK_IPAR_AUTO_SORT_A_BEFORE_OPT, 155
MSK_IPAR_AUTO_UPDATE_SOL_INFO, 155
MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE, 155
MSK_IPAR_BI_CLEAN_OPTIMIZER, 155
MSK_IPAR_BI_IGNORE_MAX_ITER, 155
MSK_IPAR_BI_IGNORE_NUM_ERROR, 156
MSK_IPAR_BI_MAX_ITERATIONS, 156

MSK_IPAR_CACHE_LICENSE, 156
MSK_IPAR_CHECK_CONVEXITY, 156
MSK_IPAR_COMPRESS_STATFILE, 156
MSK_IPAR_INFEAS_GENERIC_NAMES, 156
MSK_IPAR_INFEAS_PREFER_PRIMAL, 156
MSK_IPAR_INFEAS_REPORT_AUTO, 157
MSK_IPAR_INFEAS_REPORT_LEVEL, 157
MSK_IPAR_INTPNT_BASIS, 157
MSK_IPAR_INTPNT_DIFF_STEP, 157
MSK_IPAR_INTPNT_HOTSTART, 157
MSK_IPAR_INTPNT_MAX_ITERATIONS, 157
MSK_IPAR_INTPNT_MAX_NUM_COR, 157
MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS, 158
MSK_IPAR_INTPNT_MULTI_THREAD, 158
MSK_IPAR_INTPNT_OFF_COL_TRH, 158
MSK_IPAR_INTPNT_ORDER_METHOD, 158
MSK_IPAR_INTPNT_REGULARIZATION_USE, 158
MSK_IPAR_INTPNT_SCALING, 158
MSK_IPAR_INTPNT_SOLVE_FORM, 158
MSK_IPAR_INTPNT_STARTING_POINT, 158
MSK_IPAR_LICENSE_DEBUG, 159
MSK_IPAR_LICENSE_PAUSE_TIME, 159
MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS, 159
MSK_IPAR_LICENSE_TRH_EXPIRY_WRN, 159
MSK_IPAR_LICENSE_WAIT, 159
MSK_IPAR_LOG, 159
MSK_IPAR_LOG_ANA_PRO, 159
MSK_IPAR_LOG_BI, 160
MSK_IPAR_LOG_BI_FREQ, 160
MSK_IPAR_LOG_CHECK_CONVEXITY, 160
MSK_IPAR_LOG_CUT_SECOND_OPT, 160
MSK_IPAR_LOG_EXPAND, 160
MSK_IPAR_LOG_FACTOR, 160
MSK_IPAR_LOG_FEAS_REPAIR, 160
MSK_IPAR_LOG_FILE, 161
MSK_IPAR_LOG_HEAD, 161
MSK_IPAR_LOG_INFEAS_ANA, 161
MSK_IPAR_LOG_INTPNT, 161
MSK_IPAR_LOG_MIO, 161
MSK_IPAR_LOG_MIO_FREQ, 161
MSK_IPAR_LOG_OPTIMIZER, 161
MSK_IPAR_LOG_ORDER, 161
MSK_IPAR_LOG_PRESOLVE, 162
MSK_IPAR_LOG_RESPONSE, 162
MSK_IPAR_LOG_SENSITIVITY, 162
MSK_IPAR_LOG_SENSITIVITY_OPT, 162
MSK_IPAR_LOG_SIM, 162
MSK_IPAR_LOG_SIM_FREQ, 162
MSK_IPAR_LOG_SIM_MINOR, 162
MSK_IPAR_LOG_STORAGE, 163
MSK_IPAR_MAX_NUM_WARNINGS, 163
MSK_IPAR_MIO_BRANCH_DIR, 163
MSK_IPAR_MIO_CONSTRUCT_SOL, 163
MSK_IPAR_MIO_CUT_CLIQUE, 163
MSK_IPAR_MIO_CUT_CMIR, 163
MSK_IPAR_MIO_CUT_GMI, 163
MSK_IPAR_MIO_CUT_IMPLIED_BOUND, 164
MSK_IPAR_MIO_CUT_KNAPSACK_COVER, 164
MSK_IPAR_MIO_CUT_SELECTION_LEVEL, 164
MSK_IPAR_MIO_HEURISTIC_LEVEL, 164
MSK_IPAR_MIO_MAX_NUM_BRANCHES, 164
MSK_IPAR_MIO_MAX_NUM_RELAXS, 164
MSK_IPAR_MIO_MAX_NUM_SOLUTIONS, 164
MSK_IPAR_MIO_MODE, 165
MSK_IPAR_MIO_MT_USER_CB, 165
MSK_IPAR_MIO_NODE_OPTIMIZER, 165
MSK_IPAR_MIO_NODE_SELECTION, 165
MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE, 165
MSK_IPAR_MIO_PROBING_LEVEL, 165
MSK_IPAR_MIO_RINS_MAX_NODES, 165
MSK_IPAR_MIO_ROOT_OPTIMIZER, 166
MSK_IPAR_MIO_ROOT_REPEAT_PRESOLVE_LEVEL, 166
MSK_IPAR_MIO_VB_DETECTION_LEVEL, 166
MSK_IPAR_MT_SPINCOUNT, 166
MSK_IPAR_NUM_THREADS, 166
MSK_IPAR_OPF_MAX_TERMS_PER_LINE, 166
MSK_IPAR_OPF_WRITE_HEADER, 166
MSK_IPAR_OPF_WRITE_HINTS, 167
MSK_IPAR_OPF_WRITE_PARAMETERS, 167
MSK_IPAR_OPF_WRITE_PROBLEM, 167
MSK_IPAR_OPF_WRITE_SOL_BAS, 167
MSK_IPAR_OPF_WRITE_SOL_ITG, 167
MSK_IPAR_OPF_WRITE_SOL_ITR, 167
MSK_IPAR_OPF_WRITE_SOLUTIONS, 167
MSK_IPAR_OPTIMIZER, 168
MSK_IPAR_PARAM_READ_CASE_NAME, 168
MSK_IPAR_PARAM_READ_IGN_ERROR, 168
MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_FILL, 168
MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES, 168
MSK_IPAR_PRESOLVE_LEVEL, 168
MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH, 168
MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH, 168
MSK_IPAR_PRESOLVE_LINDEP_USE, 168
MSK_IPAR_PRESOLVE_MAX_NUM_REDUCTIONS, 169
MSK_IPAR_PRESOLVE_USE, 169
MSK_IPAR_PRIMAL_REPAIR_OPTIMIZER, 169
MSK_IPAR_READ_DATA_COMPRESSED, 169
MSK_IPAR_READ_DATA_FORMAT, 169
MSK_IPAR_READ_DEBUG, 169
MSK_IPAR_READ_KEEP_FREE_CON, 169
MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU, 169
MSK_IPAR_READ_LP_QUOTED_NAMES, 170
MSK_IPAR_READ_MPS_FORMAT, 170
MSK_IPAR_READ_MPS_WIDTH, 170
MSK_IPAR_READ_TASK_IGNORE_PARAM, 170
MSK_IPAR_SENSITIVITY_ALL, 170
MSK_IPAR_SENSITIVITY_OPTIMIZER, 170
MSK_IPAR_SENSITIVITY_TYPE, 170
MSK_IPAR_SIM_BASIS_FACTOR_USE, 170
MSK_IPAR_SIM_DEGEN, 171
MSK_IPAR_SIM_DUAL_CRASH, 171
MSK_IPAR_SIM_DUAL_PHASEONE_METHOD, 171
MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION, 171

MSK_IPAR_SIM_DUAL_SELECTION, 171
MSK_IPAR_SIM_EXPLOIT_DUPVEC, 171
MSK_IPAR_SIM_HOTSTART, 172
MSK_IPAR_SIM_HOTSTART_LU, 172
MSK_IPAR_SIM_INTEGER, 172
MSK_IPAR_SIM_MAX_ITERATIONS, 172
MSK_IPAR_SIM_MAX_NUM_SETBACKS, 172
MSK_IPAR_SIM_NON_SINGULAR, 172
MSK_IPAR_SIM_PRIMAL_CRASH, 172
MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD, 172
MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION, 173
MSK_IPAR_SIM_PRIMAL_SELECTION, 173
MSK_IPAR_SIM_REFACTOR_FREQ, 173
MSK_IPAR_SIM_REFORMULATION, 173
MSK_IPAR_SIM_SAVE_LU, 173
MSK_IPAR_SIM_SCALING, 173
MSK_IPAR_SIM_SCALING_METHOD, 173
MSK_IPAR_SIM_SOLVE_FORM, 174
MSK_IPAR_SIM_STABILITY_PRIORITY, 174
MSK_IPAR_SIM_SWITCH_OPTIMIZER, 174
MSK_IPAR_SOL_FILTER_KEEP_BASIC, 174
MSK_IPAR_SOL_FILTER_KEEP_RANGED, 174
MSK_IPAR_SOL_READ_NAME_WIDTH, 174
MSK_IPAR_SOL_READ_WIDTH, 174
MSK_IPAR_SOLUTION_CALLBACK, 174
MSK_IPAR_TIMING_LEVEL, 175
MSK_IPAR_WRITE_BAS_CONSTRAINTS, 175
MSK_IPAR_WRITE_BAS_HEAD, 175
MSK_IPAR_WRITE_BAS_VARIABLES, 175
MSK_IPAR_WRITE_DATA_COMPRESSED, 175
MSK_IPAR_WRITE_DATA_FORMAT, 175
MSK_IPAR_WRITE_DATA_PARAM, 175
MSK_IPAR_WRITE_FREE_CON, 175
MSK_IPAR_WRITE_GENERIC_NAMES, 176
MSK_IPAR_WRITE_GENERIC_NAMES_IO, 176
MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_ITEMS, 176
MSK_IPAR_WRITE_INT_CONSTRAINTS, 176
MSK_IPAR_WRITE_INT_HEAD, 176
MSK_IPAR_WRITE_INT_VARIABLES, 176
MSK_IPAR_WRITE_LP_FULL_OBJ, 176
MSK_IPAR_WRITE_LP_LINE_WIDTH, 176
MSK_IPAR_WRITE_LP_QUOTED_NAMES, 177
MSK_IPAR_WRITE_LP_STRICT_FORMAT, 177
MSK_IPAR_WRITE_LP_TERMS_PER_LINE, 177
MSK_IPAR_WRITE_MPS_FORMAT, 177
MSK_IPAR_WRITE_MPS_INT, 177
MSK_IPAR_WRITE_PRECISION, 177
MSK_IPAR_WRITE_SOL_BARVARIABLES, 177
MSK_IPAR_WRITE_SOL_CONSTRAINTS, 177
MSK_IPAR_WRITE_SOL_HEAD, 178
MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES, 178
MSK_IPAR_WRITE_SOL_VARIABLES, 178
MSK_IPAR_WRITE_TASK_INC_SOL, 178
MSK_IPAR_WRITE_XML_MODE, 178
String params, 178
MSK_SPAR_BAS_SOL_FILE_NAME, 178
MSK_SPAR_DATA_FILE_NAME, 178
MSK_SPAR_DEBUG_FILE_NAME, 178
MSK_SPAR_INT_SOL_FILE_NAME, 179
MSK_SPAR_ITR_SOL_FILE_NAME, 179
MSK_SPAR_MIO_DEBUG_STRING, 179
MSK_SPAR_PARAM_COMMENT_SIGN, 179
MSK_SPAR_PARAM_READ_FILE_NAME, 179
MSK_SPAR_PARAM_WRITE_FILE_NAME, 179
MSK_SPAR_READ_MPS_BOU_NAME, 179
MSK_SPAR_READ_MPS_OBJ_NAME, 179
MSK_SPAR_READ_MPS_RAN_NAME, 179
MSK_SPAR_READ_MPS_RHS_NAME, 179
MSK_SPAR_REMOTE_ACCESS_TOKEN, 180
MSK_SPAR_SENSITIVITY_FILE_NAME, 180
MSK_SPAR_SENSITIVITY_RES_FILE_NAME, 180
MSK_SPAR_SOL_FILTER_XC_LOW, 180
MSK_SPAR_SOL_FILTER_XC_UPR, 180
MSK_SPAR_SOL_FILTER_XX_LOW, 180
MSK_SPAR_SOL_FILTER_XX_UPR, 180
MSK_SPAR_STAT_FILE_NAME, 180
MSK_SPAR_STAT_KEY, 180
MSK_SPAR_STAT_NAME, 181
MSK_SPAR_WRITE_LP_GEN_VAR_NAME, 181

Response codes

MSK_RES_OK (ok), 192
MSK_RES_TRM_INTERNAL (trm_internal) , 192
MSK_RES_TRM_INTERNAL_STOP (trm_internal_stop) , 192
MSK_RES_TRM_MAX_ITERATIONS (trm_max_iterations) , 192
MSK_RES_TRM_MAX_NUM_SETBACKS (trm_max_num_setbacks) , 192
MSK_RES_TRM_MAX_TIME (trm_max_time) , 192
MSK_RES_TRM_MIO_NEAR_ABS_GAP (trm_mio_near_abs_gap) , 193
MSK_RES_TRM_MIO_NEAR_REL_GAP (trm_mio_near_rel_gap) , 193
MSK_RES_TRM_MIO_NUM_BRANCHES (trm_mio_num_branches) , 193
MSK_RES_TRM_MIO_NUM_RELAXS (trm_mio_num_relaxs) , 193
MSK_RES_TRM_NUM_MAX_NUM_INT_SOLUTIONS (trm_num_max_num_int_solutions) , 193
MSK_RES_TRM_NUMERICAL_PROBLEM (trm_numerical_problem) , 193
MSK_RES_TRM_OBJECTIVE_RANGE (trm_objective_range) , 193
MSK_RES_TRM_STALL (trm_stall) , 193
MSK_RES_TRM_USER_CALLBACK (trm_user_callback) , 193
MSK_RES_WRN_ANA_ALMOST_INT_BOUNDS (wrn_ana_almost_int_bounds) , 210
MSK_RES_WRN_ANA_C_ZERO (wrn_ana_c_zero) , 210
MSK_RES_WRN_ANA_CLOSE_BOUNDS (wrn_ana_close_bounds) , 211

MSK_RES_WRN_ANA_EMPTY_COLS (wrn_ana_empty_cols) , 211
MSK_RES_WRN_ANA_LARGE_BOUNDS (wrn_ana_large_bounds) , 211
MSK_RES_WRN_CONSTRUCT_INVALID_SOL_ITG (wrn_construct_invalid_sol_itg) , 211
MSK_RES_WRN_CONSTRUCT_NO_SOL_ITG (wrn_construct_no_sol_itg) , 211
MSK_RES_WRN_CONSTRUCT_SOLUTION_INFEAS (wrn_construct_solution_infeas) , 211
MSK_RES_WRN_DROPPED_NZ_QOBJ (wrn_dropped_nz_qobj) , 211
MSK_RES_WRN_DUPLICATE_BARVARIABLE_NAMES (wrn_duplicate_barvariable_names) , 211
MSK_RES_WRN_DUPLICATE_CONE_NAMES (wrn_duplicate_cone_names) , 211
MSK_RES_WRN_DUPLICATE_CONSTRAINT_NAMES (wrn_duplicate_constraint_names) , 211
MSK_RES_WRN_DUPLICATE_VARIABLE_NAMES (wrn_duplicate_variable_names) , 211
MSK_RES_WRN_ELIMINATOR_SPACE (wrn_eliminator_space) , 211
MSK_RES_WRN_EMPTY_NAME (wrn_empty_name) , 211
MSK_RES_WRN_IGNORE_INTEGER (wrn_ignore_integer) , 211
MSK_RES_WRN_INCOMPLETE_LINEAR_DEPENDENCY_CHECK (wrn_incomplete_linear_dependency_check) , 211
MSK_RES_WRN_LARGE_AIJ (wrn_large_aij) , 211
MSK_RES_WRN_LARGE_BOUND (wrn_large_bound) , 211
MSK_RES_WRN_LARGE_CJ (wrn_large_cj) , 211
MSK_RES_WRN_LARGE_CON_FX (wrn_large_con_fx) , 211
MSK_RES_WRN_LARGE_LO_BOUND (wrn_large_lo_bound) , 211
MSK_RES_WRN_LARGE_UP_BOUND (wrn_large_up_bound) , 212
MSK_RES_WRN_LICENSE_EXPIRE (wrn_license_expire) , 212
MSK_RES_WRN_LICENSE_FEATURE_EXPIRE (wrn_license_feature_expire) , 212
MSK_RES_WRN_LICENSE_SERVER (wrn_license_server) , 212
MSK_RES_WRN_LP_DROP_VARIABLE (wrn_lp_drop_variable) , 212
MSK_RES_WRN_LP_OLD_QUAD_FORMAT (wrn_lp_old_quad_format) , 212
MSK_RES_WRN_MIO_INFEASIBLE_FINAL (wrn_mio_infeasible_ nal) , 212
MSK_RES_WRN_MPS_SPLIT_BOU_VECTOR (wrn_mps_split_bou_vector) , 212
MSK_RES_WRN_MPS_SPLIT_RAN_VECTOR (wrn_mps_split_ran_vector) , 212
MSK_RES_WRN_MPS_SPLIT_RHS_VECTOR (wrn_mps_split_rhs_vector) , 212
MSK_RES_WRN_NAME_MAX_LEN (wrn_name_max_len) , 212
MSK_RES_WRN_NO_DUALIZER (wrn_no_dualizer) , 212
MSK_RES_WRN_NO_GLOBAL_OPTIMIZER (wrn_no_global_optimizer) , 212
MSK_RES_WRN_NO_NONLINEAR_FUNCTION_WRITE (wrn_no_nonlinear_function_write) , 212
MSK_RES_WRN_NZ_IN_UPR_TRI (wrn_nz_in_upr_tri) , 212
MSK_RES_WRN_OPEN_PARAM_FILE (wrn_open_param_ le) , 212
MSK_RES_WRN_PARAM_IGNORED_CMIO (wrn_param_ignored_cmio) , 212
MSK_RES_WRN_PARAM_NAME_DOU (wrn_param_name_dou) , 212
MSK_RES_WRN_PARAM_NAME_INT (wrn_param_name_int) , 212
MSK_RES_WRN_PARAM_NAME_STR (wrn_param_name_str) , 212
MSK_RES_WRN_PARAM_STR_VALUE (wrn_param_str_value) , 212
MSK_RES_WRN_PRESOLVE_OUTOFSPACE (wrn_presolve_outofspace), 213
MSK_RES_WRN_QUAD_CONES_WITH_ROOT_FIXED_AT_ZERO (wrn_quad_cones_with_root_ xed_at_zero) , 213
MSK_RES_WRN_RQUAD_CONES_WITH_ROOT_FIXED_AT_ZERO (wrn_rquad_cones_with_root_ xed_at_zero) , 213
MSK_RES_WRN_SOL_FILE_IGNORED_CON (wrn_sol_ le_ignored_con) , 213
MSK_RES_WRN_SOL_FILE_IGNORED_VAR (wrn_sol_ le_ignored_var) , 213
MSK_RES_WRN_SOL_FILTER (wrn_sol_ lter) , 213
MSK_RES_WRN_SPAR_MAX_LEN (wrn_spar_max_len) , 213
MSK_RES_WRN_SYM_MAT_LARGE (wrn_sym_mat_large) , 213
MSK_RES_WRN_TOO_FEW_BASIS_VARS (wrn_too_few_basis_vars) , 213
MSK_RES_WRN_TOO_MANY_BASIS_VARS (wrn_too_many_basis_vars) , 213
MSK_RES_WRN_UNDEF_SOL_FILE_NAME (wrn_undef_sol_ le_name) , 213
MSK_RES_WRN_USING_GENERIC_NAMES (wrn_using_generic_names) , 213
MSK_RES_WRN_WRITE_CHANGED_NAMES (wrn_write_changed_names) , 213
MSK_RES_WRN_WRITE_DISCARDED_CFIX (wrn_write_discarded_c x) , 213
MSK_RES_WRN_ZERO_AIJ (wrn_zero_aij) , 213
MSK_RES_WRN_ZEROS_IN_SPARSE_COL (wrn_zeros_in_sparse_col) , 213
MSK_RES_WRN_ZEROS_IN_SPARSE_ROW (wrn_zeros_in_sparse_row) , 213

MSK_RES_ERR_AD_INVALID_CODELIST (err_ad_invalid_codelist) , 193
MSK_RES_ERR_API_ARRAY_TOO_SMALL (err_api_array_too_small) , 193
MSK_RES_ERR_API_CB_CONNECT (err_api_cb_connect) , 193
MSK_RES_ERR_API_FATAL_ERROR (err_api_fatal_error) , 193
MSK_RES_ERR_API_INTERNAL (err_api_internal) , 193
MSK_RES_ERR_ARG_IS_TOO_LARGE (err_arg_is_too_large) , 193
MSK_RES_ERR_ARG_IS_TOO_SMALL (err_arg_is_too_small) , 194
MSK_RES_ERR_ARGUMENT_DIMENSION (err_argument_dimension) , 194
MSK_RES_ERR_ARGUMENT_IS_TOO_LARGE (err_argument_is_too_large) , 194
MSK_RES_ERR_ARGUMENT_LENNEQ (err_argument_lenneq) , 194
MSK_RES_ERR_ARGUMENT_PERM_ARRAY (err_argument_perm_array) , 194
MSK_RES_ERR_ARGUMENT_TYPE (err_argument_type) , 194
MSK_RES_ERR_BAR_VAR_DIM (err_bar_var_dim) , 194
MSK_RES_ERR_BASIS (err_basis) , 194
MSK_RES_ERR_BASIS_FACTOR (err_basis_factor) , 194
MSK_RES_ERR_BASIS_SINGULAR (err_basis_singular) , 194
MSK_RES_ERR_BLANK_NAME (err_blank_name) , 194
MSK_RES_ERR_CANNOT_CLONE_NL (err_cannot_clone_nl) , 194
MSK_RES_ERR_CANNOT_HANDLE_NL (err_cannot_handle_nl) , 194
MSK_RES_ERR_CBF_DUPLICATE_ACOORD (err_cbf_duplicate_acoord) , 194
MSK_RES_ERR_CBF_DUPLICATE_BCOORD (err_cbf_duplicate_bcoord) , 194
MSK_RES_ERR_CBF_DUPLICATE_CON (err_cbf_duplicate_con) , 194
MSK_RES_ERR_CBF_DUPLICATE_INT (err_cbf_duplicate_int) , 194
MSK_RES_ERR_CBF_DUPLICATE_OBJ (err_cbf_duplicate_obj) , 194
MSK_RES_ERR_CBF_DUPLICATE_OBJACOORD (err_cbf_duplicate_objacoord) , 194
MSK_RES_ERR_CBF_DUPLICATE_VAR (err_cbf_duplicate_var) , 194
MSK_RES_ERR_CBF_INVALID_CON_TYPE (err_cbf_invalid_con_type) , 194
MSK_RES_ERR_CBF_INVALID_DOMAIN_DIMENSION (err_cbf_invalid_domain_dimension) , 194
MSK_RES_ERR_CBF_INVALID_INT_INDEX (err_cbf_invalid_int_index) , 194
MSK_RES_ERR_CBF_INVALID_VAR_TYPE (err_cbf_invalid_var_type) , 195
MSK_RES_ERR_CBF_NO_VARIABLES (err_cbf_no_variables) , 195
MSK_RES_ERR_CBF_NO_VERSION_SPECIFIED (err_cbf_no_version_speci ed) , 195
MSK_RES_ERR_CBF_OBJ_SENSE (err_cbf_obj_sense) , 195
MSK_RES_ERR_CBF_PARSE (err_cbf_parse) , 195
MSK_RES_ERR_CBF_SYNTAX (err_cbf_syntax) , 195
MSK_RES_ERR_CBF_TOO_FEW_CONSTRAINTS (err_cbf_too_few_constraints) , 195
MSK_RES_ERR_CBF_TOO_FEW_INTS (err_cbf_too_few_ints) , 195
MSK_RES_ERR_CBF_TOO_FEW_VARIABLES (err_cbf_too_few_variables) , 195
MSK_RES_ERR_CBF_TOO_MANY_CONSTRAINTS (err_cbf_too_many_constraints) , 195
MSK_RES_ERR_CBF_TOO_MANY_INTS (err_cbf_too_many_ints) , 195
MSK_RES_ERR_CBF_TOO_MANY_VARIABLES (err_cbf_too_many_variables) , 195
MSK_RES_ERR_CBF_UNSUPPORTED (err_cbf_unsupported) , 195
MSK_RES_ERR_CON_Q_NOT_NSD (err_con_q_not_nsd) , 195
MSK_RES_ERR_CON_Q_NOT_PSD (err_con_q_not_psd) , 195
MSK_RES_ERR_CONE_INDEX (err_cone_index) , 195
MSK_RES_ERR_CONE_OVERLAP (err_cone_overlap) , 195
MSK_RES_ERR_CONE_OVERLAP_APPEND (err_cone_overlap_append) , 195
MSK_RES_ERR_CONE_REP_VAR (err_cone_rep_var) , 195
MSK_RES_ERR_CONE_SIZE (err_cone_size) , 195
MSK_RES_ERR_CONE_TYPE (err_cone_type) , 196
MSK_RES_ERR_CONE_TYPE_STR (err_cone_type_str) , 196
MSK_RES_ERR_DATA_FILE_EXT (err_data_ le_ext) , 196
MSK_RES_ERR_DUP_NAME (err_dup_name) , 196
MSK_RES_ERR_DUPLICATE_AIJ (err_duplicate_aij) , 196
MSK_RES_ERR_DUPLICATE_BARVARIABLE_NAMES (err_duplicate_barvariable_names) , 196
MSK_RES_ERR_DUPLICATE_CONE_NAMES (err_duplicate_cone_names) , 196
MSK_RES_ERR_DUPLICATE_CONSTRAINT_NAMES (err_duplicate_constraint_names) , 196
MSK_RES_ERR_DUPLICATE_VARIABLE_NAMES (err_duplicate_variable_names) , 196
MSK_RES_ERR_END_OF_FILE (err_end_of_ le) , 196
MSK_RES_ERR_FACTOR (err_factor) , 196
MSK_RES_ERR_FEASREPAIR_CANNOT_RELAX (err_feasrepair_cannot_relax) , 196

MSK_RES_ERR_FEASREPAIR_INCONSISTENT_BOUND (err_feasrepair_inconsistent_bound) , 196
MSK_RES_ERR_FEASREPAIR_SOLVING_RELAXED (err_feasrepair_solving_relaxed) , 196
MSK_RES_ERR_FILE_LICENSE (err_ le_license) , 196
MSK_RES_ERR_FILE_OPEN (err_ le_open) , 196
MSK_RES_ERR_FILE_READ (err_ le_read) , 196
MSK_RES_ERR_FILE_WRITE (err_ le_write) , 196
MSK_RES_ERR_FIRST (err_ rst) , 196
MSK_RES_ERR_FIRSTI (err_ rsti) , 196
MSK_RES_ERR_FIRSTJ (err_ rstj) , 196
MSK_RES_ERR_FIXED_BOUND_VALUES (err_ xed_bound_values) , 196
MSK_RES_ERR_FLEXLM (err_ exlm) , 197
MSK_RES_ERR_GLOBAL_INV_CONIC_PROBLEM (err_global_inv_conic_problem) , 197
MSK_RES_ERR_HUGE_AIJ (err_huge_aij) , 197
MSK_RES_ERR_HUGE_C (err_huge_c) , 197
MSK_RES_ERR_IDENTICAL_TASKS (err_identical_tasks) , 197
MSK_RES_ERR_IN_ARGUMENT (err_in_argument) , 197
MSK_RES_ERR_INDEX (err_index) , 197
MSK_RES_ERR_INDEX_ARR_IS_TOO_LARGE (err_index_arr_is_too_large) , 197
MSK_RES_ERR_INDEX_ARR_IS_TOO_SMALL (err_index_arr_is_too_small) , 197
MSK_RES_ERR_INDEX_IS_TOO_LARGE (err_index_is_too_large) , 197
MSK_RES_ERR_INDEX_IS_TOO_SMALL (err_index_is_too_small) , 197
MSK_RES_ERR_INF_DOU_INDEX (err_inf_dou_index) , 197
MSK_RES_ERR_INF_DOU_NAME (err_inf_dou_name) , 197
MSK_RES_ERR_INF_INT_INDEX (err_inf_int_index) , 197
MSK_RES_ERR_INF_INT_NAME (err_inf_int_name) , 197
MSK_RES_ERR_INF_LINT_INDEX (err_inf_lint_index) , 197
MSK_RES_ERR_INF_LINT_NAME (err_inf_lint_name) , 197
MSK_RES_ERR_INF_TYPE (err_inf_type) , 197
MSK_RES_ERR_INFEAS_UNDEFINED (err_infeas_unde ned) , 197
MSK_RES_ERR_INFINITE_BOUND (err_in nite_bound) , 197
MSK_RES_ERR_INT64_TO_INT32_CAST (err_int64_to_int32_cast) , 197
MSK_RES_ERR_INTERNAL (err_internal) , 197
MSK_RES_ERR_INTERNAL_TEST_FAILED (err_internal_test_failed) , 198
MSK_RES_ERR_INV_APTRE (err_inv_aptre) , 198
MSK_RES_ERR_INV_BK (err_inv_bk) , 198
MSK_RES_ERR_INV_BKC (err_inv_bkc) , 198
MSK_RES_ERR_INV_BKX (err_inv_bkx) , 198
MSK_RES_ERR_INV_CONE_TYPE (err_inv_cone_type) , 198
MSK_RES_ERR_INV_CONE_TYPE_STR (err_inv_cone_type_str) , 198
MSK_RES_ERR_INV_MARKI (err_inv_marki) , 198
MSK_RES_ERR_INV_MARKJ (err_inv_markj) , 198
MSK_RES_ERR_INV_NAME_ITEM (err_inv_name_item) , 198
MSK_RES_ERR_INV_NUMI (err_inv_numi) , 198
MSK_RES_ERR_INV_NUMJ (err_inv_numj) , 198
MSK_RES_ERR_INV_OPTIMIZER (err_inv_optimizer) , 198
MSK_RES_ERR_INV_PROBLEM (err_inv_problem) , 198
MSK_RES_ERR_INV_QCON_SUBI (err_inv_qcon_subi) , 198
MSK_RES_ERR_INV_QCON_SUBJ (err_inv_qcon_subj) , 198
MSK_RES_ERR_INV_QCON_SUBK (err_inv_qcon_subk) , 198
MSK_RES_ERR_INV_QCON_VAL (err_inv_qcon_val) , 198
MSK_RES_ERR_INV_QOBJ_SUBI (err_inv_qobj_subi) , 198
MSK_RES_ERR_INV_QOBJ_SUBJ (err_inv_qobj_subj) , 198
MSK_RES_ERR_INV_QOBJ_VAL (err_inv_qobj_val) , 198
MSK_RES_ERR_INV_SK (err_inv_sk) , 198
MSK_RES_ERR_INV_SK_STR (err_inv_sk_str) , 198
MSK_RES_ERR_INV_SKC (err_inv_skc) , 199
MSK_RES_ERR_INV_SKN (err_inv_skn) , 199
MSK_RES_ERR_INV_SKX (err_inv_skx) , 199
MSK_RES_ERR_INV_VAR_TYPE (err_inv_var_type) , 199
MSK_RES_ERR_INVALID_ACCMODE (err_invalid_accmode) , 199
MSK_RES_ERR_INVALID_AIJ (err_invalid_aij) , 199
MSK_RES_ERR_INVALID_AMPL_STUB (err_invalid_ampl_stub) , 199
MSK_RES_ERR_INVALID_BARVAR_NAME (err_invalid_barvar_name) , 199
MSK_RES_ERR_INVALID_COMPRESSION (err_invalid_compression) , 199
MSK_RES_ERR_INVALID_CON_NAME (err_invalid_con_name) , 199
MSK_RES_ERR_INVALID_CONE_NAME (err_invalid_cone_name) , 199
MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_CONES (err_invalid_ le_format_for_cones) , 199
MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_GENERAL_NL (err_invalid_ le_format_for_general_nl) , 199
MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_SYM_MAT (err_invalid_ le_format_for_sym_mat) , 199

MSK_RES_ERR_INVALID_FILE_NAME (err_invalid_ le_name) , 199
MSK_RES_ERR_INVALID_FORMAT_TYPE (err_invalid_format_type) , 199
MSK_RES_ERR_INVALID_IDX (err_invalid_idx) , 199
MSK_RES_ERR_INVALID_IOMODE (err_invalid_iomode) , 199
MSK_RES_ERR_INVALID_MAX_NUM (err_invalid_max_num) , 199
MSK_RES_ERR_INVALID_NAME_IN_SOL_FILE (err_invalid_name_in_sol_ le) , 199
MSK_RES_ERR_INVALID_OBJ_NAME (err_invalid_obj_name) , 199
MSK_RES_ERR_INVALID_OBJECTIVE_SENSE (err_invalid_objective_sense) , 199
MSK_RES_ERR_INVALID_PROBLEM_TYPE (err_invalid_problem_type) , 199
MSK_RES_ERR_INVALID_SOL_FILE_NAME (err_invalid_sol_ le_name) , 200
MSK_RES_ERR_INVALID_STREAM (err_invalid_stream) , 200
MSK_RES_ERR_INVALID_SURPLUS (err_invalid_surplus) , 200
MSK_RES_ERR_INVALID_SYM_MAT_DIM (err_invalid_sym_mat_dim) , 200
MSK_RES_ERR_INVALID_TASK (err_invalid_task) , 200
MSK_RES_ERR_INVALID_UTF8 (err_invalid_utf8) , 200
MSK_RES_ERR_INVALID_VAR_NAME (err_invalid_var_name) , 200
MSK_RES_ERR_INVALID_WCHAR (err_invalid_wchar) , 200
MSK_RES_ERR_INVALID_WHICHSOL (err_invalid_whichsol) , 200
MSK_RES_ERR_JSON_DATA (err_json_data) , 200
MSK_RES_ERR_JSON_FORMAT (err_json_format) , 200
MSK_RES_ERR_JSON_MISSING_DATA (err_json_missing_data) , 200
MSK_RES_ERR_JSON_NUMBER_OVERFLOW (err_json_number_over ow) , 200
MSK_RES_ERR_JSON_STRING (err_json_string) , 200
MSK_RES_ERR_JSON_SYNTAX (err_json_syntax) , 200
MSK_RES_ERR_LAST (err_last) , 200
MSK_RES_ERR_LASTI (err_lasti) , 200
MSK_RES_ERR_LASTJ (err_lastj) , 200
MSK_RES_ERR_LAU_ARG_K (err_lau_arg_k) , 200
MSK_RES_ERR_LAU_ARG_M (err_lau_arg_m) , 200
MSK_RES_ERR_LAU_ARG_N (err_lau_arg_n) , 200
MSK_RES_ERR_LAU_ARG_TRANS (err_lau_arg_trans) , 200
MSK_RES_ERR_LAU_ARG_TRANSA (err_lau_arg_transa) , 200
MSK_RES_ERR_LAU_ARG_TRANSB (err_lau_arg_transb) , 201
MSK_RES_ERR_LAU_ARG_UPLO (err_lau_arg_uplo) , 201
MSK_RES_ERR_LAU_INVALID_LOWER_TRIANGULAR_MATRIX (err_lau_invalid_lower_triangular_matrix) , 201
MSK_RES_ERR_LAU_INVALID_SPARSE_SYMMETRIC_MATRIX (err_lau_invalid_sparse_symmetric_matrix) , 201
MSK_RES_ERR_LAU_NOT_POSITIVE_DEFINITE (err_lau_not_positive_de nite) , 201
MSK_RES_ERR_LAU_SINGULAR_MATRIX (err_lau_singular_matrix) , 201
MSK_RES_ERR_LAU_UNKNOWN (err_lau_unknown) , 201
MSK_RES_ERR_LICENSE (err_license) , 201
MSK_RES_ERR_LICENSE_CANNOT_ALLOCATE (err_license_cannot_allocate) , 201
MSK_RES_ERR_LICENSE_CANNOT_CONNECT (err_license_cannot_connect) , 201
MSK_RES_ERR_LICENSE_EXPIRED (err_license_expired) , 201
MSK_RES_ERR_LICENSE_FEATURE (err_license_feature) , 201
MSK_RES_ERR_LICENSE_INVALID_HOSTID (err_license_invalid_hostid) , 201
MSK_RES_ERR_LICENSE_MAX (err_license_max) , 201
MSK_RES_ERR_LICENSE_MOSEKLM_DAEMON (err_license_moseklm_daemon) , 201
MSK_RES_ERR_LICENSE_NO_SERVER_LINE (err_license_no_server_line) , 201
MSK_RES_ERR_LICENSE_NO_SERVER_SUPPORT (err_license_no_server_support) , 201
MSK_RES_ERR_LICENSE_SERVER (err_license_server) , 201
MSK_RES_ERR_LICENSE_SERVER_VERSION (err_license_server_version) , 201
MSK_RES_ERR_LICENSE_VERSION (err_license_version) , 202
MSK_RES_ERR_LINK_FILE_DLL (err_link_ le_dll) , 202
MSK_RES_ERR_LIVING_TASKS (err_living_tasks) , 202
MSK_RES_ERR_LOWER_BOUND_IS_A_NAN (err_lower_bound_is_a_nan) , 202
MSK_RES_ERR_LP_DUP_SLACK_NAME (err_lp_dup_slack_name) , 202
MSK_RES_ERR_LP_EMPTY (err_lp_empty) , 202
MSK_RES_ERR_LP_FILE_FORMAT (err_lp_ le_format) , 202
MSK_RES_ERR_LP_FORMAT (err_lp_format) , 202
MSK_RES_ERR_LP_FREE_CONSTRAINT (err_lp_free_constraint) , 202
MSK_RES_ERR_LP_INCOMPATIBLE (err_lp_incompatible) , 202

MSK_RES_ERR_LP_INVALID_CON_NAME (err_lp_invalid_con_name) , 202
MSK_RES_ERR_LP_INVALID_VAR_NAME (err_lp_invalid_var_name) , 202
MSK_RES_ERR_LP_WRITE_CONIC_PROBLEM (err_lp_write_conic_problem) , 202
MSK_RES_ERR_LP_WRITE_GECO_PROBLEM (err_lp_write_geco_problem) , 202
MSK_RES_ERR_LU_MAX_NUM_TRIES (err_lu_max_num_tries) , 202
MSK_RES_ERR_MAX_LEN_IS_TOO_SMALL (err_max_len_is_too_small) , 202
MSK_RES_ERR_MAXNUMBARVAR (err_maxnumbarvar) , 202
MSK_RES_ERR_MAXNUMCON (err_maxnumcon) , 202
MSK_RES_ERR_MAXNUMCONE (err_maxnumcone) , 202
MSK_RES_ERR_MAXNUMQNZ (err_maxnumqnz) , 202
MSK_RES_ERR_MAXNUMVAR (err_maxnumvar) , 202
MSK_RES_ERR_MIO_INTERNAL (err_mio_internal) , 203
MSK_RES_ERR_MIO_INVALID_NODE_OPTIMIZER (err_mio_invalid_node_optimizer) , 203
MSK_RES_ERR_MIO_INVALID_ROOT_OPTIMIZER (err_mio_invalid_root_optimizer) , 203
MSK_RES_ERR_MIO_NO_OPTIMIZER (err_mio_no_optimizer) , 203
MSK_RES_ERR_MIO_NOT_LOADED (err_mio_not_loaded) , 203
MSK_RES_ERR_MISSING_LICENSE_FILE (err_missing_license_ le) , 203
MSK_RES_ERR_MIXED_CONIC_AND_NL (err_mixed_conic_and_nl) , 203
MSK_RES_ERR_MPS_CONE_OVERLAP (err_mps_cone_overlap) , 203
MSK_RES_ERR_MPS_CONE_REPEAT (err_mps_cone_repeat) , 203
MSK_RES_ERR_MPS_CONE_TYPE (err_mps_cone_type) , 203
MSK_RES_ERR_MPS_DUPLICATE_Q_ELEMENT (err_mps_duplicate_q_element) , 203
MSK_RES_ERR_MPS_FILE (err_mps_ le) , 203
MSK_RES_ERR_MPS_INV_BOUND_KEY (err_mps_inv_bound_key) , 203
MSK_RES_ERR_MPS_INV_CON_KEY (err_mps_inv_con_key) , 203
MSK_RES_ERR_MPS_INV_FIELD (err_mps_inv_ eld) , 203
MSK_RES_ERR_MPS_INV_MARKER (err_mps_inv_marker) , 203
MSK_RES_ERR_MPS_INV_SEC_NAME (err_mps_inv_sec_name) , 203
MSK_RES_ERR_MPS_INV_SEC_ORDER (err_mps_inv_sec_order) , 203
MSK_RES_ERR_MPS_INVALID_OBJ_NAME (err_mps_invalid_obj_name) , 203
MSK_RES_ERR_MPS_INVALID_OBJSENSE (err_mps_invalid_objsense) , 203
MSK_RES_ERR_MPS_MUL_CON_NAME (err_mps_mul_con_name) , 203
MSK_RES_ERR_MPS_MUL_CSEC (err_mps_mul_csec) , 203
MSK_RES_ERR_MPS_MUL_QOBJ (err_mps_mul_qobj) , 203
MSK_RES_ERR_MPS_MUL_QSEC (err_mps_mul_qsec) , 204
MSK_RES_ERR_MPS_NO_OBJECTIVE (err_mps_no_objective) , 204
MSK_RES_ERR_MPS_NON_SYMMETRIC_Q (err_mps_non_symmetric_q) , 204
MSK_RES_ERR_MPS_NULL_CON_NAME (err_mps_null_con_name) , 204
MSK_RES_ERR_MPS_NULL_VAR_NAME (err_mps_null_var_name) , 204
MSK_RES_ERR_MPS_SPLITTED_VAR (err_mps_splitted_var) , 204
MSK_RES_ERR_MPS_TAB_IN_FIELD2 (err_mps_tab_in_ eld2) , 204
MSK_RES_ERR_MPS_TAB_IN_FIELD3 (err_mps_tab_in_ eld3) , 204
MSK_RES_ERR_MPS_TAB_IN_FIELD5 (err_mps_tab_in_ eld5) , 204
MSK_RES_ERR_MPS_UNDEF_CON_NAME (err_mps_undef_con_name) , 204
MSK_RES_ERR_MPS_UNDEF_VAR_NAME (err_mps_undef_var_name) , 204
MSK_RES_ERR_MUL_A_ELEMENT (err_mul_a_element) , 204
MSK_RES_ERR_NAME_IS_NULL (err_name_is_null) , 204
MSK_RES_ERR_NAME_MAX_LEN (err_name_max_len) , 204
MSK_RES_ERR_NAN_IN_BLC (err_nan_in_blc) , 204
MSK_RES_ERR_NAN_IN_BLX (err_nan_in_blx) , 204
MSK_RES_ERR_NAN_IN_BUC (err_nan_in_buc) , 204
MSK_RES_ERR_NAN_IN_BUX (err_nan_in_bux) , 204
MSK_RES_ERR_NAN_IN_C (err_nan_in_c) , 204
MSK_RES_ERR_NAN_IN_DOUBLE_DATA (err_nan_in_double_data) , 204
MSK_RES_ERR_NEGATIVE_APPEND (err_negative_append) , 204
MSK_RES_ERR_NEGATIVE_SURPLUS (err_negative_surplus) , 204
MSK_RES_ERR_NEWER_DLL (err_newer_dll) , 204
MSK_RES_ERR_NO_BARS_FOR_SOLUTION (err_no_bars_for_solution) , 205
MSK_RES_ERR_NO_BARX_FOR_SOLUTION (err_no_barx_for_solution) , 205
MSK_RES_ERR_NO_BASIS_SOL (err_no_basis_sol) , 205
MSK_RES_ERR_NO_DUAL_FOR_ITG_SOL (err_no_dual_for_itg_sol) , 205
MSK_RES_ERR_NO_DUAL_INFEAS_CER (err_no_dual_infeas_cer) , 205

MSK_RES_ERR_NO_INIT_ENV (err_no_init_env) , 205
MSK_RES_ERR_NO_OPTIMIZER_VAR_TYPE (err_no_optimizer_var_type) , 205
MSK_RES_ERR_NO_PRIMAL_INFEAS_CER (err_no_primal_infeas_cer) , 205
MSK_RES_ERR_NO_SNX_FOR_BAS_SOL (err_no_snx_for_bas_sol) , 205
MSK_RES_ERR_NO_SOLUTION_IN_CALLBACK (err_no_solution_in_callback) , 205
MSK_RES_ERR_NON_UNIQUE_ARRAY (err_non_unique_array) , 205
MSK_RES_ERR_NONCONVEX (err_nonconvex), 205
MSK_RES_ERR_NONLINEAR_EQUALITY (err_nonlinear_equality) , 205
MSK_RES_ERR_NONLINEAR_FUNCTIONS_NOT_ALLOWED (err_nonlinear_functions_not_allowed) , 205
MSK_RES_ERR_NONLINEAR_RANGED (err_nonlinear_ranged) , 205
MSK_RES_ERR_NR_ARGUMENTS (err_nr_arguments) , 205
MSK_RES_ERR_NULL_ENV (err_null_env) , 205
MSK_RES_ERR_NULL_POINTER (err_null_pointer) , 205
MSK_RES_ERR_NULL_TASK (err_null_task) , 205
MSK_RES_ERR_NUMCONLIM (err_numconlim) , 205
MSK_RES_ERR_NUMVARLIM (err_numvarlim) , 205
MSK_RES_ERR_OBJ_Q_NOT_NSD (err_obj_q_not_nsd) , 205
MSK_RES_ERR_OBJ_Q_NOT_PSD (err_obj_q_not_psd) , 206
MSK_RES_ERR_OBJECTIVE_RANGE (err_objective_range) , 206
MSK_RES_ERR_OLDER_DLL (err_older_dll) , 206
MSK_RES_ERR_OPEN_DL (err_open_dl) , 206
MSK_RES_ERR_OPF_FORMAT (err_opf_format) , 206
MSK_RES_ERR_OPF_NEW_VARIABLE (err_opf_new_variable) , 206
MSK_RES_ERR_OPF_PREMATURE_EOF (err_opf_premature_eof) , 206
MSK_RES_ERR_OPTIMIZER_LICENSE (err_optimizer_license) , 206
MSK_RES_ERR_OVERFLOW (err_over ow) , 206
MSK_RES_ERR_PARAM_INDEX (err_param_index) , 206
MSK_RES_ERR_PARAM_IS_TOO_LARGE (err_param_is_too_large) , 206
MSK_RES_ERR_PARAM_IS_TOO_SMALL (err_param_is_too_small) , 206
MSK_RES_ERR_PARAM_NAME (err_param_name) , 206
MSK_RES_ERR_PARAM_NAME_DOU (err_param_name_dou) , 206
MSK_RES_ERR_PARAM_NAME_INT (err_param_name_int) , 206
MSK_RES_ERR_PARAM_NAME_STR (err_param_name_str) , 206
MSK_RES_ERR_PARAM_TYPE (err_param_type) , 206
MSK_RES_ERR_PARAM_VALUE_STR (err_param_value_str) , 206
MSK_RES_ERR_PLATFORM_NOT_LICENSED (err_platform_not_licensed) , 206
MSK_RES_ERR_POSTSOLVE (err_postsolve), 206
MSK_RES_ERR_PRO_ITEM (err_pro_item) , 206
MSK_RES_ERR_PROB_LICENSE (err_prob_license) , 207
MSK_RES_ERR_QCON_SUBI_TOO_LARGE (err_qcon_subi_too_large) , 207
MSK_RES_ERR_QCON_SUBI_TOO_SMALL (err_qcon_subi_too_small) , 207
MSK_RES_ERR_QCON_UPPER_TRIANGLE (err_qcon_upper_triangle) , 207
MSK_RES_ERR_QOBJ_UPPER_TRIANGLE (err_qobj_upper_triangle) , 207
MSK_RES_ERR_READ_FORMAT (err_read_format) , 207
MSK_RES_ERR_READ_LP_MISSING_END_TAG (err_read_lp_missing_end_tag) , 207
MSK_RES_ERR_READ_LP_NONEXISTING_NAME (err_read_lp_nonexisting_name) , 207
MSK_RES_ERR_REMOVE_CONE_VARIABLE (err_remove_cone_variable) , 207
MSK_RES_ERR_REPAIR_INVALID_PROBLEM (err_repair_invalid_problem) , 207
MSK_RES_ERR_REPAIR_OPTIMIZATION_FAILED (err_repair_optimization_failed) , 207
MSK_RES_ERR_SEN_BOUND_INVALID_LO (err_sen_bound_invalid_lo) , 207
MSK_RES_ERR_SEN_BOUND_INVALID_UP (err_sen_bound_invalid_up) , 207
MSK_RES_ERR_SEN_FORMAT (err_sen_format) , 207
MSK_RES_ERR_SEN_INDEX_INVALID (err_sen_index_invalid) , 207
MSK_RES_ERR_SEN_INDEX_RANGE (err_sen_index_range) , 207
MSK_RES_ERR_SEN_INVALID_REGEXP (err_sen_invalid_regexp) , 207
MSK_RES_ERR_SEN_NUMERICAL (err_sen_numerical) , 207
MSK_RES_ERR_SEN_SOLUTION_STATUS (err_sen_solution_status) , 207
MSK_RES_ERR_SEN_UNDEF_NAME (err_sen_undef_name) , 207
MSK_RES_ERR_SEN_UNHANDLED_PROBLEM_TYPE (err_sen_unhandled_problem_type) , 207
MSK_RES_ERR_SERVER_CONNECT (err_server_connect) , 207
MSK_RES_ERR_SERVER_PROTOCOL (err_server_protocol) , 208
MSK_RES_ERR_SERVER_STATUS (err_server_status) , 208
MSK_RES_ERR_SERVER_TOKEN (err_server_token) , 208

MSK_RES_ERR_SIZE_LICENSE (err_size_license), 208
MSK_RES_ERR_SIZE_LICENSE_CON (err_size_license_con), 208
MSK_RES_ERR_SIZE_LICENSE_INTVAR (err_size_license_intvar), 208
MSK_RES_ERR_SIZE_LICENSE_NUMCORES (err_size_license_numcores), 208
MSK_RES_ERR_SIZE_LICENSE_VAR (err_size_license_var), 208
MSK_RES_ERR_SOL_FILE_INVALID_NUMBER (err_sol_ le_invalid_number), 208
MSK_RES_ERR_SOLITEM (err_solitem), 208
MSK_RES_ERR_SOLVER_PROBTYPE (err_solver_probtype), 208
MSK_RES_ERR_SPACE (err_space), 208
MSK_RES_ERR_SPACE_LEAKING (err_space_leaking), 208
MSK_RES_ERR_SPACE_NO_INFO (err_space_no_info), 208
MSK_RES_ERR_SYM_MAT_DUPLICATE (err_sym_mat_duplicate), 208
MSK_RES_ERR_SYM_MAT_HUGE (err_sym_mat_huge), 208
MSK_RES_ERR_SYM_MAT_INVALID (err_sym_mat_invalid), 208
MSK_RES_ERR_SYM_MAT_INVALID_COL_INDEX (err_sym_mat_invalid_col_index), 208
MSK_RES_ERR_SYM_MAT_INVALID_ROW_INDEX (err_sym_mat_invalid_row_index), 208
MSK_RES_ERR_SYM_MAT_INVALID_VALUE (err_sym_mat_invalid_value), 208
MSK_RES_ERR_SYM_MAT_NOT_LOWER_TRINGULAR (err_sym_mat_not_lower_tringular), 208
MSK_RES_ERR_TASK_INCOMPATIBLE (err_task_incompatible), 208
MSK_RES_ERR_TASK_INVALID (err_task_invalid), 209
MSK_RES_ERR_TASK_WRITE (err_task_write), 209
MSK_RES_ERR_THREAD_COND_INIT (err_thread_cond_init), 209
MSK_RES_ERR_THREAD_CREATE (err_thread_create), 209
MSK_RES_ERR_THREAD_MUTEX_INIT (err_thread_mutex_init), 209
MSK_RES_ERR_THREAD_MUTEX_LOCK (err_thread_mutex_lock), 209
MSK_RES_ERR_THREAD_MUTEX_UNLOCK (err_thread_mutex_unlock), 209
MSK_RES_ERR_TOCONIC_CONSTR_NOT_CONIC (err_toconic_constr_not_conic), 209
MSK_RES_ERR_TOCONIC_CONSTR_Q_NOT_PSD (err_toconic_constr_q_not_psd), 209
MSK_RES_ERR_TOCONIC_CONSTRAINT_FX (err_toconic_constraint_fx), 209
MSK_RES_ERR_TOCONIC_CONSTRAINT_RA (err_toconic_constraint_ra), 209
MSK_RES_ERR_TOCONIC_OBJECTIVE_NOT_PSD (err_toconic_objective_not_psd), 209
MSK_RES_ERR_TOO_SMALL_MAX_NUM_NZ (err_too_small_max_num_nz), 209
MSK_RES_ERR_TOO_SMALL_MAXNUMANZ (err_too_small_maxnumanz), 209
MSK_RES_ERR_UNB_STEP_SIZE (err_unb_step_size), 209
MSK_RES_ERR_UNDEF_SOLUTION (err_undef_solution), 209
MSK_RES_ERR_UNDEFINED_OBJECTIVE_SENSE (err_unde ned_objective_sense), 209
MSK_RES_ERR_UNHANDLED_SOLUTION_STATUS (err_unhandled_solution_status), 210
MSK_RES_ERR_UNKNOWN (err_unknown), 210
MSK_RES_ERR_UPPER_BOUND_IS_A_NAN (err_upper_bound_is_a_nan), 210
MSK_RES_ERR_UPPER_TRIANGLE (err_upper_triangle), 210
MSK_RES_ERR_USER_FUNC_RET (err_user_func_ret), 210
MSK_RES_ERR_USER_FUNC_RET_DATA (err_user_func_ret_data), 210
MSK_RES_ERR_USER_NLO_EVAL (err_user_nlo_eval), 210
MSK_RES_ERR_USER_NLO_EVAL_HESSUBI (err_user_nlo_eval_hessubi), 210
MSK_RES_ERR_USER_NLO_EVAL_HESSUBJ (err_user_nlo_eval_hessubj), 210
MSK_RES_ERR_USER_NLO_FUNC (err_user_nlo_func), 210
MSK_RES_ERR_WHICHITEM_NOT_ALLOWED (err_whichitem_not_allowed), 210
MSK_RES_ERR_WHICHSOL (err_whichsol), 210
MSK_RES_ERR_WRITE_LP_FORMAT (err_write_lp_format), 210
MSK_RES_ERR_WRITE_LP_NON_UNIQUE_NAME (err_write_lp_non_unique_name), 210
MSK_RES_ERR_WRITE_MPS_INVALID_NAME (err_write_mps_invalid_name), 210
MSK_RES_ERR_WRITE_OPF_INVALID_VAR_NAME (err_write_opf_invalid_var_name), 210
MSK_RES_ERR_WRITING_FILE (err_writing_ le), 210
MSK_RES_ERR_XML_INVALID_PROBLEM_TYPE (err_xml_invalid_problem_type), 210
MSK_RES_ERR_Y_IS_UNDEFINED (err_y_is_unde ned), 210

Structures

Bara, 142
Barc, 141
Callback, 144
Cones, 141

Cprisen, 143
Duasen, 143
Info, 143
Names, 141
Prisen, 143
Prob, 139
Res, 140
Solution, 142
Solver_solutions, 142
Symbcon, 144
Vprisen, 143

Types

rescode, 139