# The MOSEK optimization toolbox for MATLAB manual.
## Version 7.1 (Revision 45).

www.mosek.com

ii

- Published by MOSEK ApS, Denmark.

# Contents

# Contact information

| | | |
|---|---|---|
| Phone | +45 3917 9907 | |
| Fax | +45 3917 9823 | |
| WEB | http://www.mosek.com | |
| Email | sales@mosek.com | Sales, pricing, and licensing. |
| | support@mosek.com | Technical support, questions and bug reports. |
| | info@mosek.com | Everything else. |
| Mail | `MOSEK ApS` | |
| | C/O Symbion Science Park | |
| | Fruebjergvej 3, Box 16 | |
| | 2100 Copenhagen Ō | |
| | Denmark | |

# License agreement

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

`mosek\7\license.pdf`

# Chapter 1

# Changes and new features in MOSEK

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

## 1.1  Platform support

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

## 1.2  General changes

- The interior-point optimizer has been extended to semi-definite optimization problems. Hence, MOSEK can optimize over the positive semi-definite cone.

- The network detection has been completely redesigned. MOSEK no longer try detect partial networks. The problem must be a pure primal network for the network optimizer to be used.

- The parameter iparam.objective_sense has been removed.

- The parameter iparam.intpnt_num_threads has been removed. Use the parameter iparam.num_threads instead.

- MOSEK now automatically exploit multiple CPUs i.e. the parameter iparam.num_threads is set to 0 be default. Note the amount memory that MOSEK uses grows with the number of threads employed.

5

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

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

Table 1.1: Supported platforms

- The MBT file format has been replaced by a new task format. The new format supports semi-definite optimization.

- the HTML version of the documentation is no longer included in the downloads to save space. It is still available online.

- MOSEK is more restrictive about the allowed names on variables etc. This is in particular the case when writing LP files.

- MOSEK no longer tries to detect the cache sizes and is in general less sensitive to the hardware.

- The parameter is set iparam.auto_update_sol_info is default off. In previous version it was by default on.

- The function relaxprimal has been deprecated and replaced by the function primalrepair.

## 1.3  Optimizers

### 1.3.1  Interior point optimizer

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

### 1.3.2  The simplex optimizers

- No major changes.

### 1.3.3   Mixed-integer optimizer

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

## 1.4   API changes

- Added support for semidefinite optimization.

- Some clean up has been performed implying some functions have been renamed.

## 1.5   Optimization toolbox for MATLAB

- A MOSEK equivalent of `bintprog` has been introduced.

- The functionality of the MOSEK version of `linprog` has been improved. It is now possible to employ the simplex optimizer in `linprog`.

- `mosekopt` now accepts a dense $A$ matrix.

- An new method for specification of cones that is more efficient when the problem has many cones has introduced. The old method is still allowed but is deprecated.

- Support for semidefinite optimization problems has been added to the toolbox.

## 1.6   License system

- Flexlm has been upgraded to version 11.11.

## 1.7   Other changes

- The documentation has been improved.

## 1.8   Interfaces

- Semi-definite optimization capabilities have been add to the optimizer APIs.

- A major clean up have occured in the optimizer APIs. This should have little effect for most users.

- A new object orientated interface called Fusion has been added. Fusion is available Java, MATLAB, .NET and Python.

- The AMPL command line tool has been updated to the latest version.

## 1.9   Platform changes

- 32 bit MAC OSX on Intel x86 (osx32x86) is no longer supported.

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

# Chapter 2

# Introduction

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

## 2.1    What is optimization?

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

## 2.2    Why you need the MOSEK optimization toolbox

Before solving an optimization problem data is gathered and prepared. Subsequently an optimization problem is formulated based on this data and the problem is communicated to the optimization software. Finally, when the results have been obtained, they are analyzed and interpreted. A popular software tool for these tasks is MATLAB, made by MathWorks (see http://www.mathworks.com). The MOSEK optimization solvers are well-established and succesfully deployed throughout the optimization industry, and the MOSEK toolbox provides access to the powerful MOSEK solvers from inside the MATLAB environment.

### 2.2.1    Features of the MOSEK optimization toolbox

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

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

- Solve conic quadratic and semidefinite optimization problems.

- Solve convex quadratic optimization problems.

- Handle convex quadratic constraints.

- Solve mixed-integer optimization problems, including linear, convex quadratic and conic quadratic problems.

- Solve linear least squares problems with linear constraints.

- Solve linear $\mathtt{l}_1$ and $\mathtt{l}_\infty$ norm minimization problems.

- Solve linearly constrained entropy optimization problems.

- Solve geometric programming problems (posynomial programming).

- Solve separable convex optimization problems.

- Read and write industry standard MPS files.

# Chapter 3

# Supported MATLAB versions

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

| Platform | R2009b-R2011b | R2012a | R2012b | R2013a+ |
|---|---|---|---|---|
| linux32x86 | Yes | Yes | | |
| linux64x86 | Yes | Yes | Yes | Yes |
| osx64x86 | | Yes | Yes | Yes |
| win32x86 | Yes | Yes | Yes | Yes |
| win64x86 | Yes | Yes | Yes | Yes |

Table 3.1: Supported MATLAB versions.

# Chapter 4

# Installation

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

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

## 4.1 Locating the toolbox functions

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

### 4.1.1 On Windows

If you are using Windows you should do

```
% For versions R2009b to R2011b
addpath 'c:\Program Files\mosek\7\toolbox\r2009b'

% For versions R2012a and R2012b
addpath 'c:\Program Files\mosek\7\toolbox\r2012a'

% For R2013a or newer
addpath 'c:\Program Files\mosek\7\toolbox\r2013a'
```

This assumes that you installed MOSEK at

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

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

### 4.1.2   On Linux/UNIX/MAC OS X

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

```
% For versions R2009b to R2011b
%   These versions are not supported on MAC OSX.
addpath '/home/user/mosek/7/toolbox/r2009b'

% For versions R2012a and R2012b
addpath '/home/user/mosek/7/toolbox/r2012a'

% For versions R2013a or newer
addpath '/home/user/mosek/7/toolbox/r2013a'
```

This assumes that MOSEK is installed at

```
/home/user
```

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

### 4.1.3   Keeping defaults version of Optimization toolbox functions

Setting the search path as above will overload Matlab toolbox functions such as `linprog`, `quadprog`, etc. If this is not desired, the search path can be set to

```
% For versions R2009b to R2011b
addpath 'c:\Program Files\mosek\7\toolbox\r2009bom'

% For versions R2012a and R2012b
addpath 'c:\Program Files\mosek\7\toolbox\r2012aom'

% For R2013a or newer
addpath 'c:\Program Files\mosek\7\toolbox\r2013aom'
```

on Windows, or

```
% For versions R2009b to R2011b
%   These versions are not supported on MAC OSX.
addpath '/home/user/mosek/7/toolbox/r2009bom'

% For versions R2012a and R2012b
addpath '/home/user/mosek/7/toolbox/r2012aom'

% For versions R2013a or newer
addpath '/home/user/mosek/7/toolbox/r2013aom'
```

on UNIX. This will still give access to the MOSEK toolbox, but will keep standard Matlab toolbox functions in the search path.

### 4.1.4   Permanently changing `matlabpath`

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

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

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

```
\File\Set Path
```

menu item.

## 4.2 Verifying that MOSEK works

You can verify that MOSEK works by executing

```
mosekdiag
```

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

```
Matlab version: 7.13.0.564 (R2011b)
Architecture  : GLNXA64
The mosek optimizer executed successfully from the command line:

MOSEK Version 7.0.0.78 (Build date: 2013-8-13 14:11:04)
Copyright (c) 1998-2013 MOSEK ApS, Denmark. WWW: http://mosek.com
Global optimizer version: 8.0.868.287. Global optimizer build date: Jan 21 2013 10:42:51
Barrier Solver Version 7.0.0.078,
Platform Linux 64x86 (D).

FlexLM
 Version     : 11.11
 Hostname    : gram
 Host ID     : "bc305bea244f bc305bea2450"
 Search path : /home/someuser/mosek/7/mosek.lic

Operating system variables
 MOSEKLM_LICENSE_FILE      : /home/joachim/mosekprj/generic/license/1000.lic
 LD_LIBRARY_PATH           :
   /remote/public/matlab/r2011b/sys/os/glnxa64
   /remote/public/matlab/r2011b/bin/glnxa64
   /remote/public/matlab/r2011b/extern/lib/glnxa64
   /remote/public/matlab/r2011b/sys/java/jre/glnxa64/jre/lib/amd64/native_threads
   /remote/public/matlab/r2011b/sys/java/jre/glnxa64/jre/lib/amd64/server
   /remote/public/matlab/r2011b/sys/java/jre/glnxa64/jre/lib/amd64

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

Return code - 0  [MSK_RES_OK]

mosekopt: /home/someuser/mosek/7/toolbox/r2009b/mosekopt.mexa64
mosekopt is working correctly.
MOSEK Fusion is working correctly.
```

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

## 4.3   Troubleshooting

### 4.3.1   Undefined function or variable 'mosekopt'

If you get the MATLAB error message

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

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

### 4.3.2   Invalid MEX-file

For certain versions of Microsoft Windows and MATLAB, the path to the MEX files cannot contain spaces. If you installed MOSEK in

```
 C:\Program Files\Mosek
```

and get a MATLAB error from `mosekopt`:

```
 Invalid MEX-file 'C:\Program
 Files\Mosek\7\toolbox\r2012a\mosekopt.mexw64'
```

Then try installing MOSEK in a different directory, for example

```
 C:\Users\someuser\mosek
```

### 4.3.3   Output arguments not assigned

If you encounter an error like

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

 Output argument "r" (and maybe others) not assigned during call to
 "C:\Users\someuser\mosek\7\toolbox\r2009b\mosekopt.m>mosekopt".
```

then there is most like a mismatch between 32 and 64 versions of MOSEK and MATLAB.

From MATLAB type

```
 >> which mosekopt
```

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

```
 C:\Users\someuser\mosek\7\toolbox\r2009b\mosekopt.mexw64
```

and not a MATLAB `.m` file,

```
 C:\Users\someuser\mosek\7\toolbox\r2009b\mosekopt.m
```

# Chapter 5

# Getting support and help

## 5.1   MOSEK documentation

For an overview of the available MOSEK documentation please see

```
mosek/7/docs/
```

in the distribution.

## 5.2   Additional reading

In this manual it is assumed that the reader is familiar with mathematics and in particular mathematical optimization. Some introduction to linear programming is found in books such as "Linear programming" by Chvátal [1] or "Computer Solution of Linear Programs" by Nazareth [2]. For more theoretical aspects see e.g. "Nonlinear programming: Theory and algorithms" by Bazaraa, Shetty, and Sherali [3]. Finally, the book "Model building in mathematical programming" by Williams [4] provides an excellent introduction to modeling issues in optimization.

Another useful resource is "Mathematical Programming Glossary" available at

http://glossary.computing.society.informs.org

# Chapter 6

# MOSEK / MATLAB integration

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

## 6.1   MOSEK replacements for MATLAB functions

MOSEK provides replacements for the MATLAB functions:

- `linprog`

- `quadprog`

- `lsqlin`

- `lsqnonneg`

- `bintprog`

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

## 6.2   The license system

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

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

implies that the license is released immediately after the optimizer has terminated.

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

### 6.2.1   Waiting for a free license

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

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

### 6.2.2   Using MOSEK with the Parallel Computing Toolbox

Running MOSEK with the 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 7

# A guided tour

## 7.1 Introduction

One of the big advantages of MATLAB is that it makes it very easy to do experiments and try out things without doing a lot of programming. The MOSEK optimization toolbox has been designed with this in mind. Hence, it should be very easy to solve optimization problems using MOSEK Moreover, a guided tour to the optimization toolbox has been designed to introduce the toolbox by examples. After having studyied these examples, the reader should be able to solve his or her own optimization problems without much further effort. Nevertheless, for the user interested in exploiting the toolbox to the limits, a detailed discussion and command reference are provided in the following chapters.

## 7.2 The tour starts

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

`msklpopt`

    Performs linear optimization.

`mskqpopt`

    Performs quadratic optimization.

`mskenopt`

    Performs entropy optimization.

`mskgpopt`

    Performs geometric optimization (posynomial case).

`mskscopt`

Performs separable convex optimization.

The bottom layer of the MOSEK optimization toolbox consists of one procedure named `mosekopt` This procedure provides a very flexible and powerful interface to the MOSEK optimization package. However, the price for this flexibility is a more complicated calling procedure.

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

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

`mosek\7\toolbox\examp\`

## 7.3   The MOSEK terminology

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

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

- Interior-point optimizer.

- Conic interior-point optimizer.

- Primal simplex optimizer.

- Mixed-integer optimizer.

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

## 7.4   Linear optimization

The first example is the linear optimization problem

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

### 7.4.1 Using `msklpopt`

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

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

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

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

─────────────────────────────────────[ `lo1.m` ]─────────────────────────────────────

```
1   % lo1.m
2
3   c    = [1 2 0]';
4   a    = [[1 0 1];[1 1 0]];
5   blc  = [4 1]';
6   buc  = [6 inf]';
7   blx  = sparse(3,1);
8   bux  = [];
9   [res] = msklpopt(c,a,blc,buc,blx,bux);
10  sol   = res.sol;
11
12  % Interior-point solution.
13
14  sol.itr.xx'      % x solution.
15  sol.itr.sux'     % Dual variables corresponding to buc.
16  sol.itr.slx'     % Dual variables corresponding to blx.
17
18  % Basic solution.
19
20  sol.bas.xx'      % x solution in basic solution.
```

─────────────────────────────────────────────────────────────────────────────────────

Please note that

- Infinite bounds are specified using `-inf` and `inf`. Moreover, the `bux = []` means that all upper bounds $u^x$ are plus infinite.

- The `[res] = msklpopt(c,a,blc,buc)` call implies that the lower and upper bounds on $x$ are minus and plus infinity respectively.

- The lines after the `msklpopt` call can be omitted, but the purpose of those lines is to display different parts of the solutions. The `res.sol` field contains one or more solutions. In this case both the interior-point solution (`sol.itr`) and the basic solution (`sol.bas`) are defined.

### 7.4.2 Using `mosekopt`

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

for a `mosekopt` call is

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

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

**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 (7.1) and solve the problem using `mosekopt`

──────────────────────────────────────[ lo2.m ]──────────────────────────────────────

```
1   % lo2.m
2
3   clear prob;
4
5   % Specify the c vector.
6   prob.c = [ 1 2 0]';
7
8   % Specify a in sparse format.
9   subi   = [1 2 2 1];
10  subj   = [1 1 2 3];
11  valij  = [1.0 1.0 1.0 1.0];
12
13  prob.a = sparse(subi,subj,valij);
14
15  % Specify lower bounds of the constraints.
16  prob.blc  = [4.0 1.0]';
17
18  % Specify  upper bounds of the constraints.
19  prob.buc  = [6.0 inf]';
20
21  % Specify lower bounds of the variables.
22  prob.blx  = sparse(3,1);
23
24  % Specify upper bounds of the variables.
25  prob.bux = [];   % There are no bounds.
26
27  % Perform the optimization.
28  [r,res] = mosekopt('minimize',prob);
29
30  % Show the optimal x solution.
31  res.sol.bas.xx
```

──────────────────────────────────────────────────────────────────────────────────────

Please note that

- A MATLAB structure named `prob` containing all the relevant problem data is defined.

- All fields of this structure are optional except `prob.a` which is required to be a **sparse** matrix.

- Different parts of the solution can be viewed by inspecting the solution field `res.sol`.

### 7.4.3   Using `linprog`

MOSEK also provides a `linprog` function, which is compatible with the function provided by the MATLAB toolbox.

The `linprog` functions solves a linear optimization problem:

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

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 (8.4) 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.

## 7.5   Convex quadratic optimization

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

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

In general, a quadratic optimization problem has the form

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

which for the example (7.3) implies that

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

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

### 7.5.1   Two important assumptions

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

$$Q = Q^T$$

and that $Q$ is *positive semidefinite.*A matrix is positive semidefinite if the smallest eigenvalue of the matrix is nonnegative. An alternative statement of the positive semidefinite requirement is

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

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. The MATLAB command `eig` computes all eigenvalues of a matrix.

### 7.5.2   Using `mskqpopt`

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

─────────────────────────────[ qo1.m ]─────────────────────────────

```
1   % qo1.m
2
3   % Set up Q.
4   q    = [[2 0 -1];[0 0.2 0];[-1 0 2]];
5
6   % Set up the linear part of the problem.
7   c    = [0 -1 0]';
```

```
8   a     = ones(1,3);
9   blc   = [1.0];
10  buc   = [inf];
11  blx   = sparse(3,1);
12  bux   = [];
13
14  % Optimize the problem.
15  [res] = mskqpopt(q,c,a,blc,buc,blx,bux);
16
17  % Show the primal solution.
18  res.sol.itr.xx
```

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

## 7.5.3 Using `mosekopt`

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

──────────────────────────────[ qo2.m ]──────────────────────────────
```
1   % qo2.m
2
3   clear prob;
4
5   % c vector.
6   prob.c = [0 -1 0]';
7
8   % Define the data.
9
10  % First the lower triangular part of q in the objective
11  % is specified in a sparse format. The format is:
12  %
13  %   Q(prob.qosubi(t),prob.qosubj(t)) = prob.qoval(t), t=1,...,4
14
15  prob.qosubi = [ 1   3 2    3]';
16  prob.qosubj = [ 1   1 2    3]';
17  prob.qoval  = [ 2 -1 0.2 2]';
18
19  % a, the constraint matrix
20  subi  = ones(3,1);
21  subj  = 1:3;
22  valij = ones(3,1);
23
24  prob.a = sparse(subi,subj,valij);
25
26  % Lower bounds of constraints.
27  prob.blc  = [1.0]';
28
29  % Upper bounds of constraints.
30  prob.buc  = [inf]';
31
32  % Lower bounds of variables.
33  prob.blx  = sparse(3,1);
```

```
34
35   % Upper bounds of variables.
36   prob.bux = [];    % There are no bounds.
37
38   [r,res] = mosekopt('minimize',prob);
39
40   % Display return code.
41   fprintf('Return code: %d\n',r);
42
43   % Display primal solution for the constraints.
44   res.sol.itr.xc'
45
46   % Display primal solution for the variables.
47   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

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

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

## 7.6   Conic optimization

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

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

in the problem definition where $\mathcal{C}$ is required to be a *convex cone*. The resulting class of problems is known as *conic optimization*. MOSEK can solve a subset of all conic problems and subsequently it is demonstrated how to solve this subset using the `mosekopt` toolbox function.

### 7.6.1   The conic optimization problem

A conic optimization problem has the following form

$$
\begin{array}{llcccl}
\text{minimize} & & & c^T x + c^f & & \\
\text{subject to} & l^c & \leq & Ax & \leq & u^c, \\
& l^x & \leq & x & \leq & u^x, \\
& & & x \in \mathcal{C}, &
\end{array}
\tag{7.5}
$$

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

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

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

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

Next, define

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

where $\mathcal{C}_t$ must have one of the following forms.

- $\mathbb{R}$ set:

$$\mathcal{C}_t = \{ x \in \mathbb{R}^{n^t} \}.$$

- Quadratic cone:

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

- Rotated quadratic cone:

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

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

Although the cones MOSEK can handle give rise to a limi-ted class of conic pro-blems it includes linear, quadratic, quadratically constrained optimization, and other classes of nonlinear convex optimization problems. See Section 17.2 for a discussion.

## 7.6.2 Solving an example

The problem

$$
\begin{array}{rlll}
\text{minimize} & x_4 + x_5 + x_6 & & \\
\text{subject to} & x_1 + x_2 + 2x_3 & = & 1, \\
& x_1, x_2, x_3 & \geq & 0, \\
& x_4 \geq \sqrt{x_1^2 + x_2^2}, & & \\
& 2x_5 x_6 \geq x_3^2 & &
\end{array}
\tag{7.6}
$$

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.

The following MATLAB code demonstrates how to solve the example (7.6) using MOSEK.

————————————————————————————————[ `cqo1.m` ]————————————————————————————————

```
1   % cqo1.m
2
3   clear prob;
4
5   [r, res] = mosekopt('symbcon');
6   % Specify the non-conic part of the problem.
7
8   prob.c   = [0 0 0 1 1 1];
9   prob.a   = sparse([1 1 2 0 0 0]);
10  prob.blc = 1;
11  prob.buc = 1;
12  prob.blx = [0 0 0 -inf -inf -inf];
13  prob.bux = inf*ones(6,1);
14
15  % Specify the cones.
16
17  prob.cones.type   = [res.symbcon.MSK_CT_QUAD, res.symbcon.MSK_CT_RQUAD];
18  prob.cones.sub    = [4, 1, 2, 5, 6, 3];
19  prob.cones.subptr = [1, 4];
20  % The field 'type' specifies the cone types, i.e., quadratic cone
21  % or rotated quadratic cone. The keys for the two cone types are MSK_CT_QUAD
22  % and MSK_CT_RQUAD, respectively.
23  %
24  % The fields 'sub' and 'subptr' specify the members of the cones,
25  % i.e., the above definitions imply that
26  %   x(4) >= sqrt(x(1)^2+x(2)^2) and 2 * x(5) * x(6) >= x(3)^2.
27
28  % Optimize the problem.
29
30  [r,res]=mosekopt('minimize',prob);
31
32  % Display the primal solution.
33
34  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.

## 7.6.3  Quadratic and conic optimization

The example

$$
\begin{array}{llcl}
\text{minimize} & x_1 + x_2 + x_3 & & \\
\text{subject to} & x_1^2 + x_2^2 + x_3^2 & \leq & 1, \\
& x_1 + 0.5x_2^2 + x_3 & \leq & 0.5
\end{array}
\tag{7.7}
$$

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

Indeed the first constraint is equivalent to

$$
\begin{array}{rcl}
x_4 & \geq & \sqrt{x_1^2 + x_2^2 + x_3^2}, \\
x_4 & = & 1
\end{array}
$$

where $x_4$ is a new variable. This is a quadratic cone and a linear constraint. The second constraint in (7.7) is equivalent to

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

because this implies that

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

and that

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

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

$$
x_2 = x_7
$$

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

Therefore, the example (7.7) is equivalent to the conic quadratic optimization problem

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

This problem can be solved using MOSEK as follows:

────────────────────────────────────[ cqo2.m ]────────────────────────────────────

```
1    % cqo2.m
2
3    [r, res] = mosekopt('symbcon');
4
5    % Set up the non-conic part of the problem.
6    prob          = [];
7    prob.c        = [1 1 1 0 0 0 0]';
8    prob.a        = sparse([[1 0 1 0 1 0 0];...
9                            [0 1 0 0 0 0 -1]]);
10   prob.blc      = [0.5 0];
11   prob.buc      = [0.5 0];
12   prob.blx      = [-inf -inf -inf 1 -inf 1 -inf];
13   prob.bux      = [inf   inf  inf 1  inf 1  inf];
14
15   % Set up the cone information.
16   prob.cones.type   = [res.symbcon.MSK_CT_QUAD, ...
17                        res.symbcon.MSK_CT_RQUAD];
18   prob.cones.sub    = [4, 1, 2, 3, 5, 6, 7];
19   prob.cones.subptr = [1, 5];
20
21   [r,res]       = mosekopt('minimize',prob);
22
23   % Display the solution.
24   res.sol.itr.xx'
```

────────────────────────────────────────────────────────────────────────────────────

### 7.6.4   Conic duality and the dual solution

The dual problem corresponding to the conic optimization problem (7.5) is given by

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

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

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

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

- $\mathbb{R}$ set:

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

- Quadratic cone:

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

- Rotated quadratic cone:

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

For a more detailed discussion about conic duality see Section 17.2.

#### 7.6.4.1 How to obtain the dual solution

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

──────────────────────────────[ cqo3.m ]──────────────────────────────

```
1   % cqo3.m
2
3   [r,res]=mosekopt('minimize',prob);
4
5   % Solution record.
6   res.sol
7
8   % Dual variables for lower
9   % bounds of constraints.
10  res.sol.itr.slc'
11
12  % Dual variables for upper
13  % bounds of constraints.
14  res.sol.itr.suc'
15
16  % Dual variables for lower
17  % bounds on variables.
18  res.sol.itr.slx'
19
20  % Dual variables for upper
21  % bounds on variables.
22  res.sol.itr.sux'
23
24  % Dual variables with respect
25  % to the conic constraints.
26  res.sol.itr.snx'
```

_____

### 7.6.5 Setting accuracy parameters for the conic optimizer

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

solution, if required.

_____

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

param = [];

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

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

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

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

## 7.7   Semidefinite optimization

A further generalization of conic quadratic optimization is semidefinite optimization, where we add symmetric positive semidefinite variables $\overline{X}_j \in \mathcal{S}_{r_j}^+$ of dimension $r_j$. An $n \times n$ matrix $A$ is said to be symmetric positive semidefinite if $A = A^T$ and

$$z^T A z \geq 0, \forall z \in \mathbb{R}^n,$$

and we write it equivalently as $A \succeq 0$ or $A \in \mathcal{S}_n^+$, where $\mathcal{S}_n$ ($\mathcal{S}_n^+$) is the cone of symmetric (positive semidefinite) matrices. It is easy to verify that just as the quadratic cone

$$\mathcal{Q}_n = \left\{ x \in \mathbb{R}^n \mid x_1 \geq \|x_{2:n}\| \right\}$$

specifies a cone, so does the cone of symmetric positive semidefinite matrices,

$$\mathcal{S}_n^+ = \left\{ X \in \mathbb{R}^{n \times n} \mid X = X^T, \ z^T X z \geq 0, \ \forall z \in \mathbb{R}^n \right\}$$

which warrants the notion "conic optimization". Semidefinite cones are more general that both linear and quadratic cones, however. Both linear and quadratic cones can be described using semidefinite cones, although this should be avoided in practice for efficiency reasons.

### 7.7.1   The semidefinite optimization problem

A semidefinite optimization problem is specified in MOSEK as

$$\text{minimize} \qquad \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{p} \langle \overline{C}_j, \overline{X}_j \rangle + c^f$$

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

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

$$x \in \mathcal{C}, \overline{X}_j \in \mathcal{S}_{r_j}^+, \qquad\qquad\qquad j = 1, \dots, p$$

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

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

The matrices $\overline{C}_j, \overline{A}_{ij} \in \mathcal{S}_{r_j}$ are specified in triplet format in MOSEK, using three arrays specifying the subscripts and numerical values of all nonzero elements. All matrices $\overline{C}_1, \dots, \overline{C}_p$ are then specified using four arrays `barc.subj`, `barc.subk`, `barc.subl` and `barc.val` such that

$$\left[\overline{C}_{\texttt{barc.subj(t)}}\right]_{\texttt{barc.subk(t)},\texttt{barc.subl(t)}} = \texttt{barc.val(t)}, \text{ for } t = 1, \dots, \text{length}(\texttt{barc.subj}).$$

where the dimensions $r_j$ are stored in a separate array `bardim`. Similarly all matrices $\overline{A}_{ij}$ are specified using five arrays `bara.subi`, `bara.subj`, `bara.subk`, `bara.subl`, and `bara.val` such that

$$\left[\overline{A}_{\texttt{bara.subi(t)},\texttt{bara.subj(t)}}\right]_{\texttt{bara.subk(t)},\texttt{bara.subl(t)}} = \texttt{bara.val(t)}, \text{ for } t = 1, \dots, \text{length}(\texttt{bara.subj}).$$

Since all $\overline{C}_j, \overline{A}_{ij}$ are assumed to be symmetric, only their lower triangular parts are specified.

Some attention must be paid when formulating linear constraints involving semidefinite matrices. A common mistake is not to consider that, being all $\overline{C}_j, \overline{A}_{ij}$ symmetric, their off-diagonal entries are counted twice. Indeed in that case we can write

$$\langle \overline{A}, \overline{X} \rangle := \sum_{i=1}^{p} \overline{A}_{ii} \overline{X}_{ii} + 2 \sum_{i=1}^{p} \sum_{j=i+1}^{p} \overline{A}_{ij} \overline{X}_{ij},$$

and hence the contribution of each off-diagonal element to the linear constraint is double.

For instance, let's consider $\overline{X} \in \mathcal{S}_3^+$ and a constraint of the form $\overline{X}_{12} = 1$. Introducing a symmetric matrix

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

we write the constraint as

$$\langle A, \overline{X} \rangle = \overline{X}_{12} + \overline{X}_{21} = 2\overline{X}_{12} = 2.$$

Otherwise, we could use

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

and rewrite the constraint as

$$\langle A, \overline{X} \rangle = 0.5(\overline{X}_{12} + \overline{X}_{21}) = \overline{X}_{12} = 1.$$

## 7.7.2 Solving an example

The problem

$$
\begin{array}{rllll}
\text{minimize} & \left\langle \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}, \overline{X} \right\rangle + x_1 & & & \\
\text{subject to} & \left\langle \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \overline{X} \right\rangle + x_1 & = & 1, \\
& \left\langle \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \overline{X} \right\rangle + x_2 + x_3 & = & 1/2, \\
& x_1 \geq \sqrt{x_2^2 + x_3^2}, & & \\
& \overline{X} \succeq 0, & &
\end{array}
\tag{7.8}
$$

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

$$\overline{X} = \begin{bmatrix} \overline{x}_{11} & \overline{x}_{11} & \overline{x}_{31} \\ \overline{x}_{21} & \overline{x}_{22} & \overline{x}_{32} \\ \overline{x}_{31} & \overline{x}_{32} & \overline{x}_{33} \end{bmatrix} \in \mathcal{S}_3^+,$$

and a conic quadratic variable $(x_1, x_2, x_3) \in \mathcal{Q}_3$. The objective is to minimize

$$2(\overline{x}_{11} + \overline{x}_{21} + \overline{x}_{22} + \overline{x}_{32} + \overline{x}_{33}) + x_1,$$

subject to the two linear constraints

$$\overline{x}_{11} + \overline{x}_{22} + \overline{x}_{33} + x_1 = 1,$$

and

$$\overline{x}_{11} + \overline{x}_{22} + \overline{x}_{33} + 2(\overline{x}_{21} + \overline{x}_{31} + \overline{x}_{32}) + x_2 + x_3 = 1/2.$$

The following MATLAB code demonstrates how to solve this problem using MOSEK.

─────────────────────────────────────[ sdo1.m ]─────────────────────────────────────

```
1   function sdo1()
2   % sdo1.m
3   %
4   % Solves the mixed semidefinite and conic quadratic optimization problem
5   %
6   %   minimize     Tr [2, 1, 0; 1, 2, 1; 0, 1, 2]*X + x(1)
7   %
8   %   subject to  Tr [1, 0, 0; 0, 1, 0; 0, 0, 1]*X + x(1)                = 1
9   %               Tr [1, 1, 1; 1, 1, 1; 1, 1, 1]*X        + x(2) + x(3) = 0.5
10  %               X>=0,   x(1) >= sqrt(x(2)^2 + x(3)^2)
11
12  [r, res] = mosekopt('symbcon');
13
14  prob.c       = [1, 0, 0];
15
16  prob.bardim    = [3];
17  prob.barc.subj = [1, 1, 1, 1, 1];
18  prob.barc.subk = [1, 2, 2, 3, 3];
19  prob.barc.subl = [1, 1, 2, 2, 3];
20  prob.barc.val  = [2.0, 1.0, 2.0, 1.0, 2.0];
21
22  prob.blc = [1, 0.5];
23  prob.buc = [1, 0.5];
24
25  prob.a         = sparse([1, 2, 2], [1, 2, 3], [1, 1, 1], 2, 3);
26  prob.bara.subi = [1, 1, 1, 2, 2, 2, 2, 2, 2];
27  prob.bara.subj = [1, 1, 1, 1, 1, 1, 1, 1, 1];
28  prob.bara.subk = [1, 2, 3, 1, 2, 3, 2, 3, 3];
29  prob.bara.subl = [1, 2, 3, 1, 1, 1, 2, 2, 3];
30  prob.bara.val  = [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0];
31
32  prob.cones.type   = [res.symbcon.MSK_CT_QUAD];
33  prob.cones.sub    = [1, 2, 3];
34  prob.cones.subptr = [1];
35
36  [r,res] = mosekopt('minimize info',prob);
37
38  X = zeros(3);
39  X([1,2,3,5,6,9]) = res.sol.itr.barx;
40  X = X + tril(X,-1)';
41
42  x = res.sol.itr.xx;
```

────────────────────────────────────────────────────────────────────────────────────

The solution $x$ is returned in `res.sol.itr.xx` and the numerical values of $\overline{X}_j$ are returned in `res.sol.barx`; the lower triangular part of each $\overline{X}_j$ is stacked column-by-column into an array, and

each array is then concatenated forming a single array `res.sol.itr.barx` representing $\overline{X}_1, \ldots, \overline{X}_p$. Similarly, the dual semidefinite variables $\overline{S}_j$ are recovered through `res.sol.itr.bars`.

### 7.7.3  Linear matrix inequalities

Linear matrix inequalities are affine matrix-valued functions of $z \in \mathbb{R}^n$,

$$F(z) = F_0 + z_1 F_1 + \ldots + z_n F_n \succeq 0 \tag{7.9}$$

where the coefficient $F_i \in \mathcal{S}_m^+$ are symmetric matrices (of order $m$). It is convinient to think of linear matrix inequalities as dual constraints; if a problem contains only linear matrix inequalities, then we essentially have a problem specified in dual form. If we have a problem with both semidefinite affine equalities and linear matrix inequalities, we can convert it one with only affine equality constraints. For example, the linear matrix inequality

$$F(z) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + z_1 \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + z_2 \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} + z_3 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \succeq 0$$

can be rewritten as

$$X = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{21} & x_{22} & x_{32} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \succeq 0, \quad x_{11} = x_{22} = x_{33} = 1,$$

where $z_1 = x_{21}$, $z_2 = x_{31}$ and $z_3 = x_{32}$.

## 7.8  Quadratically constrained optimization

In section (7.6) a quadratically constrained optimization problem was solved using the conic optimizer. It is also possible to solve such a problem directly. An example of such an optimization problem is

$$\begin{array}{rrcl} \text{minimize} & x_1 + x_2 + x_3 \\ \text{subject to} & x_1^2 + x_2^2 + x_3^2 & \leq & 1, \\ & x_1 + 0.1 x_2^2 + x_3 & \leq & 0.5. \end{array}$$

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

─────────────────────────────[ qco1.m ]─────────────────────────────

```
1   % qco1.m
2
3   clear prob;
4
5   % Specify the linear objective terms.
6   prob.c      = ones(3,1);
7
8   % Specify the quadratic terms of the constraints.
```

```
 9   prob.qcsubk = [1    1    1    2  ]';
10   prob.qcsubi = [1    2    3    2  ]';
11   prob.qcsubj = [1    2    3    2  ]';
12   prob.qcval  = [2.0 2.0 2.0 0.2]';
13
14   % Specify the linear constraint matrix
15   prob.a      = [sparse(1,3);sparse([1 0 1])];
16
17   prob.buc    = [1 0.5]';
18
19   [r,res]     = mosekopt('minimize',prob);
20
21   % Display the solution.
22   fprintf('\nx:');
23   fprintf(' %-.4e',res.sol.itr.xx');
24   fprintf('\n||x||: %-.4e',norm(res.sol.itr.xx));
```

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

$$Q^{\texttt{qcsubk(t)}}_{\texttt{qcsubi(t)},\texttt{qcsubj(t)}} = \texttt{qcval(t)}, \text{for} t = 1, \ldots, \text{length}(\texttt{qcsubk})$$

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

# 7.9 Linear least squares and related norm minimization problems

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

$$\text{minimize} \|Fx - b\| \tag{7.10}$$

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

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

## 7.9.1 The case of the 2 norm

Initially let us focus on the 2 norm. In this case (7.10) is identical to the quadratic optimization problem

$$\text{minimize} (1/2)x^T F^T F x + (1/2)b^T b - b^T F x \tag{7.11}$$

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 (7.11) is solved using `mosekopt`
In the example the problem data is read from a file, then data for the problem (7.11) is constructed
and finally the problem is solved.

$$\text{———————————————————}[\,\texttt{nrm1.m}\,]\text{———————————————————}$$

```
1    % nrm1.m
2
3    % Read data from 'afiro.mps'.
4    [r,res] = mosekopt('read(afiro.mps)');
5
6    % Get data for the problem
7    %                 minimize ||f x - b||_2
8    f = res.prob.a';
9    b = res.prob.c;
10
11   % Solve the problem
12   %                 minimize 0.5 xf'fx+0.5*b'*b-(f'*b)'*x
13
14   % Clear prob
15   clear prob;
16
17   % Compute the fixed term in the objective.
18   prob.cfix = 0.5*b'*b
19
20   % Create the linear objective terms
21   prob.c = -f'*b;
22
23   % Create the quadratic terms. Please note that only the lower triangular
24   % part of f'*f is used.
25   [prob.qosubi,prob.qosubj,prob.qoval] = find(sparse(tril(f'*f)))
26
27   % Obtain the matrix dimensions.
28   [m,n]   = size(f);
29
30   % Specify a.
31   prob.a  = sparse(0,n);
32
33   [r,res] = mosekopt('minimize',prob);
34
35   % The optimality conditions are f'*(f x - b) = 0.
36   % Check if they are satisfied:
37
38   fprintf('\nnorm(f^T(fx-b)): %e',norm(f'*(f*res.sol.itr.xx-b)));
```

Often the $x$ variables must be within some bounds or satisfy some additional linear constraints. These
requirements can easily be incorporated into the problem (7.11). E.g. the constraint $\|x\|_\infty \leq 1$ can
be modeled as follows:

$$\text{———————————————————}[\,\texttt{nrm2.m}\,]\text{———————————————————}$$

```
1    % nrm2.m. Continuation of nrm1.m.
2
3    % Assume that the same objective should be
4    % minimized subject to -1 <= x <= 1
5
6    prob.blx = -ones(n,1);
7    prob.bux = ones(n,1);
```

```
8
9    [r,res] = mosekopt('minimize',prob);
10
11   % Check if the solution is feasible.
12   norm(res.sol.itr.xx,inf)
```

## 7.9.2 The case of the infinity norm

In some applications of the norm minimization problem (7.10) it is better to use the infinity norm than the 2 norm. However, the problem (7.10) stated as an infinity norm problem is equivalent to the linear optimization problem

$$
\begin{array}{rlrl}
\text{minimize} & \tau & & \\
\text{subject to} & Fx + \tau e - b & \geq & 0, \\
& Fx - \tau e - b & \leq & 0,
\end{array} \tag{7.12}
$$

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

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

and hence at optimum

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

holds.

The problem (7.12) is straightforward to solve.

──────────────────────────────[ nrm3.m ]──────────────────────────────
```
1    % nrm3.m. Continuation of nrm1.m.
2
3    % Let x(n+1) play the role as tau, then the problem is
4    % solved as follows.
5
6    clear prob;
7
8    prob.c   = sparse(n+1,1,1.0,n+1,1);
9    prob.a   = [[f,ones(m,1)];[f,-ones(m,1)]];
10   prob.blc = [b              ; -inf*ones(m,1)];
11   prob.buc = [inf*ones(m,1); b             ];
12
13   [r,res]  = mosekopt('minimize',prob);
14
15   % The optimal objective value is given by:
16   norm(f*res.sol.itr.xx(1:n)-b,inf)
```

## 7.9.3 The case of the 1-norm

By definition, for the 1-norm we have that

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

Therefore, the norm minimization problem can be formulated as follows

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

which in turn is equivalent to

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{m} t_i \\
\text{subject to} \quad & f_{i:}x - b_i \quad \leq \quad t_i, \quad i = 1, \ldots, m, \\
& -(f_{i:}x - b_i) \quad \leq \quad t_i, \quad i = 1, \ldots, m.
\end{aligned}
\tag{7.13}
$$

The reader should verify that this is really the case.

In matrix notation this problem can be expressed as follows

$$
\begin{aligned}
\text{minimize} \quad & e^T t \\
\text{subject to} \quad & Fx - te \quad \leq \quad b, \\
& Fx + te \quad \geq \quad b,
\end{aligned}
\tag{7.14}
$$

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

────────────────────────────────[ nrm4.m ]────────────────────────────────

```
1   % nrm4.m. Continuation of nrm1.m.
2
3   % Let x(n:(m+n)) play the role as t. Now,
4   % the problem can be solved as follows
5
6   clear prob;
7
8   prob.c   = [sparse(n,1)    ; ones(m,1)];
9   prob.a   = [[f,-speye(m)] ; [f,speye(m)]];
10  prob.blc = [-inf*ones(m,1); b];
11  prob.buc = [b             ; inf*ones(m,1)];
12
13  [r,res]  = mosekopt('minimize',prob);
14
15  % The optimal objective value is given by:
16  norm(f*res.sol.itr.xx(1:n)-b,1)
```
────────────────────────────────────────────────────────────────────────────

### 7.9.3.1   A better formulation

It is possible to improve upon the formulation of the problem (7.13). Indeed problem (7.13) is equivalent to

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{m} t_i \\
\text{subject to} \quad & f_{i:}x - b_i - t_i + v_i \;\; = \;\; 0, \quad i = 1, \ldots, m, \\
& -(f_{i:}x - b_i) - t_i \;\; \leq \;\; 0, \quad i = 1, \ldots, m, \\
& v_i \geq 0, \qquad\qquad\qquad\;\; i = 1, \ldots, m.
\end{aligned}
\tag{7.15}
$$

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

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{m} (f_{i:}x - b_i + v_i) \\
\text{subject to} \quad & -2(f_{i:}x - b_i) - v_i \;\; \leq \;\; 0, \quad i = 1, \ldots, m, \\
& v_i \geq 0, \qquad\qquad\qquad\;\; i = 1, \ldots, m.
\end{aligned}
\tag{7.16}
$$

Please note that this problem has only half the number of general constraints than problem (7.13) since we have replaced constraints of the general form

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

with simpler constraints

$$
v_i \geq 0
$$

which MOSEK treats in a special and highly efficient way. Furthermore MOSEK stores only the non-zeros in the coefficient matrix of the constraints. This implies that the problem (7.16) is likely to require much less space than the problem (7.15).

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

## 7.10  Compatibility with MATLAB Optimization Toolbox

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

`linprog`

Solves linear optimization problems.

`quadprog`

Solves quadratic optimization problems.

`lsqlin`

Minimizes a least-squares objective with linear constraints.

`lsqnnonneg`

Minimizes a least-squares objective with nonnegativity constraints.

bintprog

    Solves a linear optimization problem with binary variables.

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 Chapter 8. Their implementations all use `mosekopt`, so unless compatibility with the MATLAB Optimization Toolbox is a neccessity, using `mosekopt` directly is recommended.

The functions `mskoptimget` and `mskoptimset` are only largely 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.

## 7.11   More about solving linear least squares problems

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

$$
\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\|Fx - f\|_2^2 \\
\text{subject to} \quad & Ax = b, \\
& l^x \leq x \leq u^x,
\end{aligned}
\tag{7.17}
$$

where $F$ and $A$ are matrices and the remaining quantities are vectors. $x$ is the vector of decision variables. The problem (7.17) as stated is a convex quadratic optimization problem and can be solved as such.

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

$$
\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\|z\|_2^2 \\
\text{subject to} \quad & Ax = b, \\
& Fx - z = f, \\
& l^x \leq x \leq u^x.
\end{aligned}
\tag{7.18}
$$

Please note that a number of new constraints and variables has been introduced which of course seems to be disadvantageous but on the other hand the Hessian of the objective in problem (7.18) is much sparser than in problem (7.17). Frequently this turns out to be more important for the computational efficiency and therefore the latter formulation is usually the better one.

If $F$ has many more rows than columns, then formulation (7.18) is not attractive whereas the corresponding dual problem is. Using the duality theory outlined in Section 17.5.1 we obtain the dual problem

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

which can be simplified to

$$
\begin{aligned}
\text{maximize} \quad & b^T y + f^T z \\
& + (l^x)^T s_l^x + (u^x)^T s_u^x \\
& - \frac{1}{2} \|z\|_2^2 \\
\text{subject to} \quad & A^T y + F^T z + s_l^x - s_u^x = 0, \\
& s_l^x, s_u^x \geq 0
\end{aligned}
\tag{7.19}
$$

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

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

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

Given our assumptions the dual problem (7.19) will have much fewer constraints than the primal problem (7.18); in general, the fewer constraints a problem contains, the more efficient MOSEK tends to be. A question is: If the dual problem (7.19) is solved instead of the primal problem (7.18), how is the optimal $x$ solution obtained? It turns out that the dual variables corresponding to the constraint

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

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

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

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

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

───────────────────────[ nrm5.m ]───────────────────────

```
1   % nrm5.m
2
3   % Read data from a file.
4   [rcode,res] = mosekopt('read(lsqpd.mps) echo(0)');
5
6   % Define the problem data.
```

```matlab
7    F          = res.prob.a;
8    f          = res.prob.blc;
9    blx        = res.prob.blx;
10   bux        = [];
11
12   % In this case there are no linear constraints
13   % First we solve the primal problem:
14   %
15   % minimize    0.5|| z ||^2
16   % subject to F x - z = f
17   %            l <= x <= u
18
19   % Note that m>>n
20   [m,n]      = size(F);
21
22   prob       = [];
23
24   prob.qosubi = n+(1:m);
25   prob.qosubj = n+(1:m);
26   prob.qoval  = ones(m,1);
27   prob.a      = [F,-speye(m,m)];
28   prob.blc    = f;
29   prob.buc    = f;
30   prob.blx    = [blx;-inf*ones(m,1)];
31   prob.bux    = bux;
32
33
34   fprintf('m=%d  n=%d\n',m,n);
35
36   fprintf('First try\n');
37
38   tic
39   [rcode,res] = mosekopt('minimize echo(0)',prob);
40
41   % Display the solution time.
42   fprintf('Time            : %-.2f\n',toc);
43
44   try
45     % x solution:
46     x = res.sol.itr.xx;
47
48     % objective value:
49     fprintf('Objective value: %-6e\n',norm(F*x(1:n)-f)^2);
50
51     % Check feasibility.
52     fprintf('Feasibility    : %-6e\n',min(x(1:n)-blx(1:n)));
53   catch
54     fprintf('MSKERROR: Could not get solution')
55   end
56
57   % Clear prob.
58   prob=[];
59
60   %
61   % Next, we solve the dual problem.
62
63   % Index of lower bounds that are finite:
64   lfin       = find(blx>-inf);
```

```
65
66   % Index of upper bounds that are finite:
67   ufin         = find(bux<inf);
68
69   prob.qosubi = 1:m;
70   prob.qosubj = 1:m;
71   prob.qoval  = -ones(m,1);
72   prob.c      = [f;blx(lfin);-bux(ufin)];
73   prob.a      = [F',...
74                   sparse(lfin,(1:length(lfin))',...
75                          ones(length(lfin),1),...
76                          n,length(lfin)),...
77                   sparse(ufin,(1:length(ufin))',...
78                          -ones(length(ufin),1),...
79                          n,length(ufin))];
80   prob.blc    = sparse(n,1);
81   prob.buc    = sparse(n,1);
82   prob.blx    = [-inf*ones(m,1);...
83                   sparse(length(lfin)+length(ufin),1)];
84   prob.bux    = [];
85
86   fprintf('\n\nSecond try\n');
87   tic
88   [rcode,res] = mosekopt('maximize echo(0)',prob);
89
90   % Display the solution time.
91   fprintf('Time           : %-.2f\n',toc);
92
93   try
94     % x solution:
95     x = res.sol.itr.y;
96
97     % objective value:
98     fprintf('Objective value: %-6e\n',...
99             norm(F*x(1:n)-f)^2);
100
101    % Check feasibility.
102    fprintf('Feasibility    : %-6e\n',...
103            min(x(1:n)-blx(1:n)));
104  catch
105    fprintf('MSKERROR: Could not get solution')
106  end
```

Here is the output produced:

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

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

Both formulations produced a strictly feasible solution having the same objective value. Moreover,

using the dual formulation leads to a reduction in the solution time by about a factor 5: In this case we can conclude that the dual formulation is far superior to the primal formulation of the problem.

### 7.11.1   Using conic optimization on linear least squares problems

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

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

is equivalent to

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

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

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

which can be reduced to

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

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.

## 7.12  Entropy optimization

### 7.12.1  Using `mskenopt`

An entropy optimization problem has the following form

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

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

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

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

---

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

---

## 7.13  Geometric optimization

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

$$\begin{array}{ll}
\text{minimize} & \sum_{k \in J_0} c_k \prod_{j=1} t_j^{a_{kj}} \\
\text{subject to} & \sum_{k \in J_i} c_k \prod_{j=1} t_j^{a_{kj}} \quad \leq \quad 1, \quad i = 1, \ldots, m, \\
& t > 0,
\end{array} \qquad (7.20)$$

where it is assumed that

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

and if $i \neq j$, then

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

Hence, $A$ is a $T \times n$ matrix and $c$ is a vector of length $t$. In general, the problem (7.20) is very hard to solve, but the posynomial case where

$$c > 0$$

is relatively easy. Using the variable transformation

$$t_j = e^{x_j} \tag{7.21}$$

we obtain the problem

$$
\begin{aligned}
\text{minimize} \quad & \sum_{k \in J_0} c_k e^{a_{k:} x} \\
\text{subject to} \quad & \sum_{k \in J_i} c_k e^{a_{k:} x} \quad \leq \quad 1, \quad i = 1, \dots, m,
\end{aligned}
\tag{7.22}
$$

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

$$
\begin{aligned}
\text{minimize} \quad & \log(\sum_{k \in J_0} c_k e^{a_{k:} x}) \\
\text{subject to} \quad & \log(\sum_{k \in J_i} c_k e^{a_{k:} x}) \quad \leq \quad \log(1), \quad i = 1, \dots, m,
\end{aligned}
\tag{7.23}
$$

which is also a convex optimization problem since `log` is strictly increasing. Hence, the problem (7.23) can be solved by MOSEK For further details about geometric optimization we refer the reader to [3].

### 7.13.1   Using `mskgpopt`

MOSEK cannot handle a geometric optimization problem directly, but the transformation (7.23) can be solved using the MOSEK optimization toolbox function `mskgpopt` Please note that the solution to the transformed problem can easily be converted into a solution to the original geometric optimization problem using relation (7.21).

Subsequently, we will use the example

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

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

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

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 $\text{map}_k$ equals zero, the $k$th term belongs to the objective function, otherwise it belongs to the $\text{map}_k$th constraint.

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

──────────────────────────────────────[ go1.m ]──────────────────────────────────────

```
1   % go1.m
2
3   c     = [40 20 40 1/3 4/3]';
4   a     = sparse([[-1  -0.5  -1];[1 0 1];...
5                   [1 1 1];[-2 -2 0];[0 0.5 -1]]);
6   map   = [0 0 0 1 1]';
7   [res] = mskgpopt(c,a,map);
8
9   fprintf('\nPrimal optimal solution to original gp:');
10  fprintf(' %e',exp(res.sol.itr.xx));
11  fprintf('\n\n');
12
13  % Compute the optimal objective value and
14  % the constraint activities.
15  v = c.*exp(a*res.sol.itr.xx);
16
17  % Add appropriate terms together.
18  f = sparse(map+1,1:5,ones(size(map)))*v;
19
20  % First objective value. Then constraint values.
21  fprintf('Objective value: %e\n',log(f(1)));
22  fprintf('Constraint values:');
23  fprintf(' %e',log(f(2:end)));
24  fprintf('\n\n');
25
26  % Dual multipliers (should be negative)
27  fprintf('Dual variables (should be negative):');
28  fprintf(' %e',res.sol.itr.y);
29  fprintf('\n\n');
```
──────────────────────────────────────────────────────────────────────────────────

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

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

or equivalently

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

### 7.13.2   Comments

#### 7.13.2.1   Solving large scale problems

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

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

where

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

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

```
help sparse
```

in MATLAB.

#### 7.13.2.2   Preprocessing tip

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

#### 7.13.2.3   Reading and writing problems to a file

The functions `mskgpread` and `mskgpwri` can used to read and write geometric programming problems to file, see the Command Reference Chap. 8.

## 7.14   Separable convex optimization

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

$$
\begin{array}{llccc}
\text{minimize} & & f(x) + c^T x & & \\
\text{subject to} & & g(x) + Ax - x^c & = & 0, \\
& l^c \leq & x^c & \leq & u^c, \\
& l^x \leq & x & \leq & u^x,
\end{array}
\tag{7.24}
$$

where

- $m$ is the number of constraints.

- $n$ is the number of decision variables.

- $x \in \mathbb{R}^n$ is a vector of decision variables.

- $x^c \in \mathbb{R}^m$ is a vector of slack variables.

- $c \in \mathbb{R}^n$ is the linear part of the objective function.

- $A \in \mathbb{R}^{m \times n}$ is the constraint matrix.

- $l^c \in \mathbb{R}^m$ is the lower limit on the activity for the constraints.

- $u^c \in \mathbb{R}^m$ is the upper limit on the activity for the constraints.

- $l^x \in \mathbb{R}^n$ is the lower limit on the activity for the variables.

- $u^x \in \mathbb{R}^n$ is the upper limit on the activity for the variables.

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

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

This implies that the $i$th constraint essentially has the form

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

when the $x_i^c$ variable has been eliminated.

The problem (7.24) must satisfy the three important requirements:

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

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

  and

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

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

- Differentiability: All functions should be twice differentiable for all $x_j$ satisfying

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

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

- Convexity:  The problem should be a convex optimization problem.  See Section 17.5 for a discussion of this requirement.

### 7.14.1   Using `mskscopt`

Subsequently, we will use the following example

$$
\begin{array}{llcl}
\text{minimize} & x_1 - \ln(x_1 + 2x_2) & & \\
\text{subject to} & x_1^2 + x_2^2 & \leq & 1
\end{array}
\tag{7.25}
$$

to demonstrate solving a convex separable optimization problem using the MOSEK optimization toolbox function `mskscopt` First, note that the problem (7.25) is not a separable optimization problem due to the fact that the logarithmic term in objective is not a function of a single variable. However, by introducing one additional constraint and variable the problem can be made separable as follows

$$
\begin{array}{llcl}
\text{minimize} & x_1 - \ln(x_3) & & \\
\text{subject to} & x_1^2 + x_2^2 & \leq & 1, \\
& x_1 + 2x_2 - x_3 & = & 0, \\
& x_3 \geq 0. & &
\end{array}
\tag{7.26}
$$

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

$$x_3 > 0.$$

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

When using the `mskscopt` function to solve problem (7.26), the linear part of the problem, such as a $c$ and $A$, is specified as usual using MATLAB vectors and matrices. However, the nonlinear functions must be specified using five arrays which in the case of problem (7.26) can have the form

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

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

| opr(k) | opri(k) | oprj(k) | oprf(k) | oprg(k) | function |
|--------|---------|---------|---------|---------|----------|
| ent | i | j | f | (not used) | $fx_j\ln(x_j)$ |
| exp | i | j | f | g | $fe^{gx_j}$ |
| log | i | j | f | (not used) | $f\ln(x_j)$ |
| pow | i | j | f | g | $fx_j^g$ |

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

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

All the elements for solving a separable convex nonlinear optimization problem have now been discussed and therefore we will conclude this section by showing the MATLAB code that will solve the example problem (7.26).

──────────────────────────────[ sco1.m ]──────────────────────────────

```matlab
1   % sco1.m
2
3   % Specify the linear part of the problem.
4
5   c          = [1;0;0];
6   a          = sparse([[0 0 0];[1 2 -1]]);
7   blc        = [-inf; 0];
8   buc        = [1;0];
9   blx        = [-inf;-inf;0];
10
11  % Specify the nonlinear part.
12
13  opr        = ['log'; 'pow'; 'pow'];
14  opri       = [0;      1;     1    ];
15  oprj       = [3;      1;     2    ];
16  oprf       = [-1;     1;     1    ];
17  oprg       = [0;      2;     2    ];
18
19  % Call the optimizer.
20  % Note that bux is an optional parameter which should be added if the variables
21  % have an upper bound.
22
23  [res]      = mskscopt(opr,opri,oprj,oprf,oprg,c,a,blc,buc,blx);
24
25
26  % Print the solution.
27  res.sol.itr.xx
```

─────────────────────────────────────────────────────────────────────────────


# 7.15   Mixed-integer optimization


Up until now it has been assumed that the variables in an optimization problem are continuous. Hence, it has been assumed that any value between the bounds on a variable is feasible. In many cases this is not a valid assumption because some variables are integer-constrained. E.g. a variable may denote

the number of persons assigned to a given job and it may not be possible to assign a fractional person.

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

### 7.15.1   A linear mixed-integer example

Using the example

$$
\begin{array}{lrcl}
\text{minimize} & -2x_1 - 3x_2 & & \\
\text{subject to} & 195x_1 + 273x_2 & \leq & 1365, \\
& 4x_1 + 40x_2 & \leq & 140, \\
& x_1 \leq 4, & & \\
& x_1, x_2 \geq 0, & & \text{and integer}
\end{array}
\tag{7.27}
$$

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

——————————————————————————[ milo1.m ]——————————————————————————
```matlab
1   % milo1.m
2
3   % Specify the linear problem data as if
4   % the problem is a linear optimization
5   % problem.
6
7   clear prob
8   prob.c        = [-2 -3];
9   prob.a        = sparse([[195 273];[4 40]]);
10  prob.blc      = -[inf inf];
11  prob.buc      = [1365 140];
12  prob.blx      = [0 0];
13  prob.bux      = [4 inf];
14
15  % Specify indexes of variables that are integer
16  % constrained.
17
18  prob.ints.sub = [1 2];
19
20  % Optimize the problem.
21  [r,res] = mosekopt('minimize',prob);
22
23  try
24    % Display the optimal solution.
25    res.sol.int
26    res.sol.int.xx'
27  catch
28    fprintf('MSKERROR: Could not get solution')
29  end
```
————————————————————————————————————————————————————————————————

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

- The `prob.ints.sub` field is used to specify the indexes of the variables that are integer-constrained.

- The optimal integer solution is returned in the `res.sol.int` MATLAB structure.

MOSEK also provides a wrapper for the `bintprog` function found in the MATLAB optimization toolbox. This function solves linear problems with binary variables only (i.e., without linear variables); see the reference section for details.

## 7.15.2 A conic quadratic mixed-integer example

The problem

$$
\begin{array}{rll}
\text{minimize} & x_5 + x_6 & \\
\text{subject to} & x_1 + x_2 + x_3 + x_4 & = & 1, \\
& x_1, x_2, x_3, x_4 & \geq & 0, \\
& x_5 \geq \sqrt{x_1^2 + x_3^2}, & & \\
& x_6 \geq \sqrt{x_2^2 + x_4^2} & & \\
& x_5, x_6 & & \text{integer}
\end{array}
\tag{7.28}
$$

is an example of a mixed-integer conic quadratic optimization problem. It is similar to example (7.6), with the exception that $x_5$ and $x_6$ are restricted to integers.

The following MATLAB code demonstrates how to solve the example (7.6) using MOSEK

────────────────────────────────────[`micqo1.m`]────────────────────────────────────
```matlab
1  % micqo1.m
2
3  clear prob;
4
5  [r, res] = mosekopt('symbcon');
6
7  % Specify the non-confic part of the problem.
8
9  prob.c   = [0 0 0 0 1 1];
10  prob.a   = sparse([1 1 1 1 0 0]);
11  prob.blc = 1;
12  prob.buc = 1;
13  prob.blx = [0 0 0 0 -inf -inf];
14  prob.bux = inf*ones(6,1);
15
16  % Specify the cones.%
17  prob.cones.type   = [res.symbcon.MSK_CT_QUAD, res.symbcon.MSK_CT_QUAD];
18  prob.cones.sub    = [5, 3, 1, 6, 2, 4];
19  prob.cones.subptr = [1, 4];
20
21  % indices of integer variables
22  prob.ints.sub = [5 6];
23
24  % Optimize the problem.
25
26  [r,res]=mosekopt('minimize',prob);
27
28  % Display the primal solution.
29
30  res.sol.int.xx'
```
─────────────────────────────────────────────────────────────────────────────────────

### 7.15.3    Speeding up the solution of a mixed-integer problem

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

How to obtain a good problem formulation is beyond the scope of this section and the reader is referred to [5]. However, two methods for assisting the mixed-integer optimizer are discussed subsequently.

#### 7.15.3.1    Specifying an initial feasible solution

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

Consider the problem:

$$
\begin{array}{lll}
\text{maximize} & 7x_0 + 10x_1 + x_2 + 5x_3 & \\
\text{subject to} & x_0 + x_1 + x_2 + x_3 & \leq \quad 2.5 \\
& x_3 \geq 0 & \\
& x_0, x_1, x_2 \geq 0 & \text{and integer,}
\end{array}
\tag{7.29}
$$

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

$$x_0 = 0, x_1 = 2, x_2 = 0, x_3 = 0.5$$

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

```
                                ─[ milo2.m ]─
1   % milo2.m
2
3   clear prob
4   clear param
5   [r,res]       = mosekopt('symbcon');
6   sc            = res.symbcon;
7
8
9   prob.c        = [7 10 1 5];
10  prob.a        = sparse([1 1 1 1 ]);
11  prob.blc      = -[inf];
12  prob.buc      = [2.5];
13  prob.blx      = [0 0 0 0];
14  prob.bux      = [inf inf inf inf];
15  prob.ints.sub = [1 2 3];
16
17  prob.sol.int.xx = [0 2 0 0.5]';
18
19  % Optionally set status keys too.
20  % prob.sol.int.skx = [sc.MSK_SK_SUPBAS;sc.MSK_SK_SUPBAS;...
21  %                     sc.MSK_SK_SUPBAS;sc.MSK_SK_BAS]
22  % prob.sol.int.skc = [sc.MSK_SK_UPR]
23
```

```
24   [r,res] = mosekopt('maximize',prob);
25
26   try
27     % Display the optimal solution.
28     res.sol.int.xx'
29   catch
30     fprintf('MSKERROR: Could not get solution')
31   end
```

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

Suppose it is known that $x_0 = 0, x_1 = 2, x_2 = 0$ are candidates for good integer values to our problem, then the following example demonstrates how to optimize the problem (7.28) using a feasible starting solution generated from the integer values as $x_0 = 0, x_1 = 2, x_2 = 0$.

───────────────────────────────────────[ milo3.m ]───────────────────────────────────────

```
1    % milo3.m
2
3    [r,res]       = mosekopt('symbcon');
4    sc            = res.symbcon;
5
6    clear prob
7
8    prob.c        = [7 10 1 5];
9    prob.a        = sparse([1 1 1 1 ]);
10   prob.blc      = -[inf];
11   prob.buc      = [2.5];
12   prob.blx      = [0 0 0 0];
13   prob.bux      = [inf inf inf inf];
14   prob.ints.sub = [1 2 3];
15
16   % Values for the integer variables are specified.
17   prob.sol.int.xx  = [0 2 0 0]';
18
19   % Tell Mosek to construct a feasible solution from a given integer
20   % value.
21   param.MSK_IPAR_MIO_CONSTRUCT_SOL = sc.MSK_ON;
22
23   [r,res] = mosekopt('maximize',prob,param);
24
25   try
26     % Display the optimal solution.
27     res.sol.int.xx'
28   catch
29     fprintf('MSKERROR: Could not get solution')
30   end
```

────────────────────────────────────────────────────────────────────────────────────────

## 7.16   Sensitivity analysis

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

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

Consider the problem:

Minimize

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

subject to

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

The example below demonstrate how sensitivity analysis can answer questions of the type: What happens to the optimal solution if we decrease the upper bound of the first constraint with 1? For more information on sensitivity analysis see Chapter 16.

──────────────────────────────[ sensitivity2.m ]──────────────────────────────

```
1    % sensitivity2.m
2
3    % Setup problem data.
4    clear prob
5    prob.a = sparse([1,    1,    0,    0,    0,    0,    0;
6                     0,    0,    1,    1,    0,    0,    0;
7                     0,    0,    0,    0,    1,    1,    1;
8                     1,    0,    0,    0,    1,    0,    0;
9                     0,    1,    0,    0,    0,    0,    0;
10                    0,    0,    1,    0,    0,    1,    0;
11                    0,    0,    0,    1,    0,    0,    1]);
12
13   prob.c =  [1,2,5,2,1,2,1];
14   prob.blc = [-Inf,-Inf,-Inf,800,100,500, 500];
15   prob.buc =[400,1200,1000,800,100,500,500];
16   prob.bux(1:7) = Inf;
17   prob.blx(1:7) = 0;
18
19   % Analyze upper bound of constraint 1.
20   prob.prisen.cons.subu = [1];
21
```

```
22   [r,res] = mosekopt('minimize echo(0)',prob);
23   fprintf ('Optimal objective value: %e\n',prob.c * res.sol.bas.xx  );
24   fprintf('Sensitivity results for constraint 1:');
25   res.prisen.cons
26
27   % If we change the upper bound of constraint 1 with a
28   % value v in [res.prisen.cons.lr_bu(1),res.prisen.cons.rr_bu(1)]
29   % then the optimal objective changes with - v * ls_bu(0)
30   % e.g. changing prob.buc(1) with -1
31   prob.buc(1) =  prob.buc(1) - 1;
32   new_sol_predicted = prob.c * res.sol.bas.xx  + 1 * res.prisen.cons.ls_bu(1);
33   fprintf ('New optimal objective after changing bound predicted to:%e\n', ...
34           new_sol_predicted);
35   [r,res] = mosekopt('minimize echo(0)',prob);
36   fprintf ('New optimal objective value: %e\n',prob.c * res.sol.bas.xx  );
```

The output from running the example is given below:

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

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


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

## 7.17  Inspecting a problem

The problem analyzer (discussed in detail in Sec. 14.1) provides useful diagnostics about an optimization problem, and is a quick way to verify that a model has been specified correctly. For example, executing the command

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

will generate a report looking like

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


-----------------------------------------------------------------------------

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

```
        [1, 10]               1

--------------------------------------------------------------------------------

Constraint matrix A has
27 rows (constraints)
32 columns (variables)
83 (9.60648%) nonzero entries (coefficients)

Row nonzeros, A_i
range: min A_i: 1 (3.125%)    max A_i: 9 (28.125%)
distrib:        A_i        rows       rows%        acc%
                  1           2        7.41        7.41
                  2          16       59.26       66.67
              [3, 7]          8       29.63       96.30
              [8, 9]          1        3.70      100.00

Column nonzeros, A|j
range: min A|j: 1 (3.7037%)    max A|j: 4 (14.8148%)
distrib:        A|j        cols       cols%        acc%
                  1           1        3.12        3.12
                  2          21       65.62       68.75
              [3, 4]         10       31.25      100.00

A nonzeros, A(ij)
range: min |A(ij)|: 0.107000    max |A(ij)|: 2.42900
distrib:        A(ij)      coeffs
            [0.107, 1)         17
             [1, 2.43]         66

--------------------------------------------------------------------------------

Constraint bounds, lb <= Ax <= ub
distrib:          |b|             lbs             ubs
                    0               7              20
            [10, 100)               1               3
           [100, 1000]                              4

Variable bounds, lb <= x <= ub
distrib:          |b|             lbs             ubs
                                    0              32
```

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

## 7.18   The solutions

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

```
 res.sol
```

structure, which has one or more of the subfields

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

```
 res.sol.int  % Integer  solution.
```

The interior (point) solution is an arbitrary optimal solution which is computed using the interior-point optimizer. The basic solution is available only for linear problems and is produced by the simplex optimizer or the basis identification process which is an add-on to the interior-point optimizer. Finally, the integer solution is available only for problems having integer-constrained variables and is computed using the integer optimizer. Each of the three solutions may contain one or more of the following subfields:

`.prosta`

> Problem status. See Appendix MSKprostae.

`.solsta`

> Solution status. See Appendix MSKsolstae.

`.skc`

> Constraint status keys. See Tablen 7.1 below.

`.skx`

> Variable status keys. See Table 7.1 below.

`.xc`

> Constraint activities.

`.xx`

> Variable activities.

`.y`

> Identical to `-.slc+.suc`.

`.slc`

> Dual variables corresponding to lower constraint bounds.

`.suc`

> Dual variables corresponding to upper constraint bounds.

`.slx`

> Dual variables corresponding to lower variable bounds.

`.sux`

> Dual variables corresponding to upper variable bounds.

`.snx`

> Dual variables corresponding to the conic constraints.

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

Table 7.1: Constraint and variable status keys.

### 7.18.1   The constraint and variable status keys

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

In Table 7.1 the possible values for the status keys are shown accompanied with an interpretation of the key. By default the constraint and variable status keys are reported using string codes but it is easy to have MOSEK report the numeric codes instead. Indeed in the example

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

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

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

the status keys are represented using numeric codes.

## 7.19   Viewing the task information

In MOSEK the optimization problem and the related instructions with respect to the optimization process are called an optimization task or for short a task. Whenever MOSEK performs operations on a task it stores information in the task information database. Examples of information that is stored are the number of interior-point iterations performed to solve the problem and time spent doing the optimization.

All the items stored in the task information database are listed in Appendixes MSKdinfiteme and MSKiinfiteme. It is possible to see the whole or part of the task information database from within MATLAB.

---

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

% View one item
res.info.MSK_IINF_INTPNT_ITER

% View the whole database
res.info
```

---

## 7.20 Inspecting and setting parameters

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

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

We will not describe the purpose of each parameter here but instead refer the reader to Appendix 23 where all the parameters are presented in detail.

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

---

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

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

param    = [];

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

% Alternatively you can use
%
%  param.MSK_IPAR_INTPNT_BASIS   = 'MSK_OFF';
%

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

% Perform optimization using the
% modified parameters.

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

---

## 7.21   Advanced start (hot-start)

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

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

### 7.21.1   Some examples using hot-start

Using the example

$$
\begin{array}{lrcccc}
\text{minimize} & & & x_1 + 2x_2 & & \\
\text{subject to} & 4 & \leq & x_1 + x_3 & \leq & 6, \\
& 1 & \leq & x_1 + x_2, & & \\
& & & 0 \leq x_1, x_2, x_3 & &
\end{array}
\tag{7.30}
$$

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

──────────────────────────────[ advs1.m ]──────────────────────────────

```
1   % advs1.m
2
3   clear prob param bas
4
5   % Specify an initial basic solution.
6   bas.skc      = ['LL';'LL'];
7   bas.skx      = ['BS';'LL';'BS'];
8   bas.xc       = [4 1]';
9   bas.xx       = [1 3 0]';
10
11  prob.sol.bas = bas;
12
13  % Specify the problem data.
14  prob.c       = [ 1 2 0]';
15  subi         = [1 2 2 1];
16  subj         = [1 1 2 3];
17  valij        = [1.0 1.0 1.0 1.0];
18  prob.a       = sparse(subi,subj,valij);
19  prob.blc     = [4.0 1.0]';
20  prob.buc     = [6.0 inf]';
21  prob.blx     = sparse(3,1);
22  prob.bux     = [];
23
24  % Use the primal simplex optimizer.
25  param.MSK_IPAR_OPTIMIZER = 'MSK_OPTIMIZER_PRIMAL_SIMPLEX';
```

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

_____

Some comments:

- In the example the dual solution is defined. This is acceptable because the primal simplex optimizer is used for the reoptimization and it does not exploit a dual solution. In the future MOSEK will also contain a dual simplex optimizer and if that optimizer is used, it will be important that a "good" dual solution is specified.

- The status keys `bas.skc` and `bas.skx` must contain only the entries BS, EQ, LL, UL, and SB. Moreover, e.g. EQ must be specified only for a fixed constraint or variable. LL and UL can be used only for a variable that has a finite lower or upper bound respectively.

- The number of constraints and variables defined to be basic must correspond exactly to the number of constraints, i.e. the row dimension of $A$.

### 7.21.2  Adding a new variable

Next, assume that the problem

$$
\begin{array}{lrcccr}
\text{minimize} & & & x_1 + 2x_2 - x_4 & & \\
\text{subject to} & 4 & \leq & x_1 + x_3 + x_4 & \leq & 6, \\
& 1 & \leq & x_1 + x_2, & & \\
& & & 0 \leq x_1, x_2, x_3, x_4. & &
\end{array}
\tag{7.31}
$$

should be solved. It is identical to the problem (7.30) except that a new variable $x_4$ has been added. In continuation of the previous example this problem can be solved as follows (using hot-start):

─────────────────────────────────[ advs2.m ]─────────────────────────────────
```
1   % advs2.m. Continuation of advs1.m.
2
3   prob.c       = [prob.c;-1.0];
4   prob.a       = [prob.a,sparse([1.0 0.0]')];
5   prob.blx     = sparse(4,1);
6
7   % Reuse the old optimal basic solution.
8   bas          = res.sol.bas;
9
10  % Add to the status key.
11  bas.skx      = [res.sol.bas.skx;'LL'];
12
13  % The new variable is at it lower bound.
14  bas.xx       = [res.sol.bas.xx;0.0];
15  bas.slx      = [res.sol.bas.slx;0.0];
16  bas.sux      = [res.sol.bas.sux;0.0];
17
18  prob.sol.bas = bas;
19
20  [rcode,res]  = mosekopt('minimize',prob,param);
21
22  % The new primal optimal solution
```

```
23   res.sol.bas.xx'
```

### 7.21.3 Fixing a variable

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

─────────────────────────────[ advs3.m ]─────────────────────────────
```
1    % advs3.m. Continuation of advs2.m.
2
3    prob.blx(4)  = 1;
4    prob.bux     = [inf inf inf 1]';
5
6    % Reuse the basis.
7    prob.sol.bas = res.sol.bas;
8
9    [rcode,res]  = mosekopt('minimize',prob,param);
10
11   % Display the optimal solution.
12   res.sol.bas.xx'
```

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

### 7.21.4 Adding a new constraint

Now, assume that the constraint

$$x_1 + x_2 \geq 2$$

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

─────────────────────────────[ advs4.m ]─────────────────────────────
```
1    % advs4.m. A continuation of advs3.m.
2
3    % Modify the problem.
4    prob.a       = [prob.a;sparse([1.0 1.0 0.0 0.0])];
5    prob.blc     = [prob.blc;2.0];
6    prob.buc     = [prob.buc;inf];
7
8    % Obtain the previous optimal basis.
9    bas          = res.sol.bas;
10
11   % Set the solution to the modified problem.
12   bas.skc      = [bas.skc;'BS'];
13   bas.xc       = [bas.xc;bas.xx(1)+bas.xx(2)];
14   bas.y        = [bas.y;0.0];
15   bas.slc      = [bas.slc;0.0];
16   bas.suc      = [bas.suc;0.0];
17
```

```
18   % Reuse the basis.
19   prob.sol.bas = bas;
20
21   % Reoptimize.
22   [rcode,res]  = mosekopt('minimize',prob,param);
23
24   res.sol.bas.xx'
```

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

### 7.21.5 Removing a constraint

We can remove a constraint in two ways:

- Set the bounds for the constraint to $+\infty$ or $-\infty$, as appropriate.

- Remove the corresponding row from `prob.a`, `prob.c`, `prob.blc` and `prob.buc` and update the basis.

In the following example we use the latter approach to again remove the constraint $x_1 + x_2 \geq 2$.

──────────────────────────────[ advs5.m ]──────────────────────────────

```
1    % advs5.m. A continuation of advs4.m.
2
3    % Modify the problem.
4    prob.a       = prob.a(1:end-1,:);
5    prob.blc     = prob.blc(1:end-1);
6    prob.buc     = prob.buc(1:end-1);
7
8    % Obtain the previous optimal basis.
9    bas          = res.sol.bas;
10
11   % Set the solution to the modified problem.
12   bas.skc      = bas.skc(1:end-1,:);
13   bas.xc       = bas.xc(1:end-1);
14   bas.y        = bas.y(1:end-1);
15   bas.slc      = bas.slc(1:end-1);
16   bas.suc      = bas.suc(1:end-1);
17
18   % Reuse the basis.
19   prob.sol.bas = bas;
20
21   % Reoptimize.
22   [rcode,res]  = mosekopt('minimize',prob,param);
23
24   res.sol.bas.xx'
```

───────────────────────────────────────────────────────────────────────

### 7.21.6 Removing a variable

Similarly we can remove a variable in two ways:

- Fixing the variable to zero.

- Remove the corresponding row from `prob.a`, `prob.blx` and `prob.bux` and update the basis.

The following example uses the latter approach to remove $x_4$.

─────────────────────────────[ advs6.m ]─────────────────────────────
```
1   % advs6.m. A continuation of advs5.m.
2
3   % Modify the problem.
4   prob.c      = prob.c(1:end-1);
5   prob.a      = prob.a(:,1:end-1);
6   prob.blx    = prob.blx(1:end-1);
7   prob.bux    = prob.bux(1:end-1);
8
9   % Obtain the previous optimal basis.
10  bas         = res.sol.bas;
11
12  % Set the solution to the modified problem.
13  bas.xx      = bas.xx(1:end-1);
14  bas.skx     = bas.skx(1:end-1,:);
15  bas.slx     = bas.slx(1:end-1);
16  bas.sux     = bas.sux(1:end-1);
17
18  % Reuse the basis.
19  prob.sol.bas = bas;
20
21  % Reoptimize.
22  [rcode,res]  = mosekopt('minimize',prob,param);
23
24  res.sol.bas.xx'
```
─────────────────────────────────────────────────────────────────────


### 7.21.7   Using numeric values to represent status key codes

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

─────────────────────────────[ sk1.m ]─────────────────────────────
```
1   % sk1.m
2
3   % Obtain all symbolic constants
4   % defined in MOSEK.
5
6   clear prob bas;
7
8   [r,res]  = mosekopt('symbcon');
9   sc       = res.symbcon;
10
11  % Specify an initial basic solution.
12  % Please note that symbolic constants are used.
13  % I.e. sc.MSK_SK_LOW instead of 4.
14  bas.skc     = [sc.MSK_SK_LOW;sc.MSK_SK_LOW];
15  bas.skx     = [sc.MSK_SK_BAS;sc.MSK_SK_LOW;sc.MSK_SK_BAS];
```

```
16   bas.xc      = [4 1]';
17   bas.xx      = [1 3 0]';
18   prob.sol.bas = bas;
19
20   % Specify the problem data.
21   prob.c  = [ 1 2 0]';
22   subi    = [1 2 2 1];
23   subj    = [1 1 2 3];
24   valij   = [1.0 1.0 1.0 1.0];
25   prob.a  = sparse(subi,subj,valij);
26   prob.blc = [4.0 1.0]';
27   prob.buc = [6.0 inf]';
28   prob.blx = sparse(3,1);
29   prob.bux = [];
30
31   % Use the primal simplex optimizer.
32   clear param;
33   param.MSK_IPAR_OPTIMIZER = sc.MSK_OPTIMIZER_PRIMAL_SIMPLEX;
34
35   [r,res] = mosekopt('minimize statuskeys(1)',prob,param)
36
37   % Status keys will be numeric now i.e.
38
39   res.sol.bas.skc'
40
41   % is a vector of numeric values.
```

Please note that using the commands

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

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

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

These two lines are in fact equivalent to

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

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

## 7.22   Using names

In MOSEK it is possible to give the objective, each constraint, each variable, and each cone a name. In general such names are not really needed except in connection with reading and writing MPS files. See Section 7.23 for details.

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

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

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

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

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

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

### 7.22.1   Blanks in names

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

## 7.23   MPS and other file formats

An industry standard format for storing linear optimization problems in an ASCII file is the so-called MPS format. For readers not familiar with the MPS format a specification of the MPS format supported by MOSEK can be seen in Appendix 18.

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

### 7.23.1   Reading an MPS file

It is possible to use `mosekopt` to read an MPS file containing the problem data. In this case `mosekopt` reads data from an MPS file and returns both the problem data and the optimal solution, if required. Assume that `afiro.mps` is the MPS file from which `mosekopt` should read the problem data, then this task is performed using the command

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

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

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

will display the $c$-vector.

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

```
% All names.
prob.names

% Constraint names.
prob.names.con
```

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

─────────────────────────────────[ `mpsrd.m` ]─────────────────────────────────

```
1   % mpsrd.m
2
3   % Read data from the file wp12-20.mps.
4
5   [r,res] = mosekopt('read(wp12-20.mps)');
6
7   % Looking at the problem data
8   prob = res.prob;
9   clear res;
10
11  % Form the quadratic term in the objective.
12  q = sparse(prob.qosubi,prob.qosubj,prob.qoval);
13
14  % Get a graphical picture.
15  spy(q) % Notice that only the lower triangular part is defined.
```

─────────────────────────────────────────────────────────────────────────────

## 7.23.2   Writing a MPS files

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

```
% Write the data defined by prob to an MPS file
% named datafile.mps
mosekopt('write(datafile.mps)',prob);
```

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

## 7.23.3   Other file formats

In addition to MPS files, MOSEK supports the following file formats.

LP file format

  Uses an `.lp` extension. See Chapter 19 for details.

OPF format

  Uses an `.opf` extension. See Chapter 20 for details.

TASK file format

  Uses a `.task` extension. See Chapter 21 for details.

XML (OSiL) file format

> Uses an `.xml` extension. See Chapter 22 for details.

The format is determined by the file extension, for example, an `.opf` file can be written as

```
mosekopt('write(problem.opf)');
```

## 7.24   User call-back functions

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

### 7.24.1   Log printing via call-back function

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

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

forces `mosekopt` to not print log information — the string `echo(0)` indicates that no output should be printed during optimization. A high value in the `echo(n)` command, e.g. `echo(3)`, forces MOSEK to display more log information. It is possible to redirect the MOSEK log printing almost anywhere using a user-defined log call-back function. It works as follows. Create an m-file to handle the log output, similar to:

─────────────────────────────[ myprint.m ]─────────────────────────────
```
1   function myprint(handle,str)
2   % handle: Is user defined data structure
3   % str   : Is a log string.
4   %
5
6   fprintf(handle,'%s',str);
```
────────────────────────────────────────────────────────────────────────

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

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

────────────────────────────────────────────────────────────────────────
```
%
% In this example the MOSEK log info
% should be printed to the screen and to a file named
% mosek.log.
%

fid                = fopen('mosek.log','wt');
callback.log       = 'myprint';
callback.loghandle = fid;
```

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

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


## 7.24.2   The iteration call-back function

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

The iteration call-back function has the following form:

─────────────────────────────────────[ `myiter.m` ]─────────────────────────────────────
```
1   function [r] = myiter(handle,where,info)
2   % handle: Is a user-defined data structure.
3   % where : Is an integer indicating from where in the optimization
4   %         process the callback was invoked.
5   % info  : A MATLAB structure containing information about the state of the
6   %         optimization.
7
8   r = 0;   % r should always be assigned a value.
9
10  if handle.symbcon.MSK_CALLBACK_BEGIN_INTPNT==where
11      fprintf('Interior point optimizer started\n');
12  end
13
14
15  % Print primal objective
16  fprintf('Interior-point primal obj.: %e\n',info.MSK_DINF_INTPNT_PRIMAL_OBJ);
17
18  % Terminate when cputime > handle.maxtime
19  if info.MSK_DINF_INTPNT_TIME > handle.maxtime
20          r = 1;
21  else
22      r = 0;
23  end
24
25
26  if handle.symbcon. MSK_CALLBACK_END_INTPNT==where
27      fprintf('Interior-point optimizer terminated\n');
28  end
```
_____

The function accepts three arguments: The first argument, `handle`, is a user-defined MATLAB structure, the second argument, `where`, indicates from where in the optimization process the call-back was invoked and the third argument, `info`, is a structure containing information about the process. For details about `info` see Section 8.1.9. If the function returns a non-zero value, MOSEK will terminate the optimization process immediately.

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

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

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

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

## 7.25   The license system

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

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

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

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

## 7.26   Caveats using the MATLAB compiler

When using MOSEK with the MATLAB compiler it is necessary to perform some manual operations,
in order to make sure that the compiler does not get confused by possible conflicts between `mosekopt.m`
and the corresponding `.mex` file.

Since `mosekopt.m` is only provided to make help information available from the MATLAB command
line, it can be safely (temporary) removed.

We therefore advise to:

- temporary remove `mosekopt.m` before compilation

- copy the mosek `.mex` file to the directory with MATLAB binary files

- copy back the `mosekopt.m` file back after compilation

# Chapter 8

# Command reference

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

## 8.1 Data structures

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

### 8.1.1 `prob`

Description:

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

Subfields:

>    `.names`
>    >    A MATLAB structure which contains the problem name, the name of the objective, and so forth. See Section 8.1.2.
>
>    `.qosubi`
>    >    $i$ subscript for element $q_{ij}^o$ in $Q^o$. See (8.3).
>
>    `.qosubj`
>    >    $j$ subscript for element $q_{ij}^o$ in $Q^o$. See (8.3).
>
>    `.qoval`
>    >    Numerical value for element $q_{ij}^o$ in $Q^o$. See (8.3).

`.qcsubk`

   $k$ subscript for element $q_{ij}^p$ in $Q^p$. See (8.4).

`.qcsubi`

   $i$ subscript for element $q_{ij}^p$ in $Q^p$. See (8.4).

`.qcsubj`

   $j$ subscript for element $q_{ij}^p$ in $Q^p$. See (8.4).

`.qcval`

   Numerical value for element $q_{ij}^p$ in $Q^p$. See (8.4).

`.c`

   Linear term in the objective.

`.a`

   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`

   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`

   A list with the dimensions of the semidefinite variables.

`.barc`

   A MATLAB structure for specifying $\overline{C}_j$. The structure has the following subfields:

   `.subj`

   `.subk`

   `.subl`

   `.val`

   See Section 8.1.4 for details on this structure.

`.bara`

   A MATLAB structure for specifying $\overline{A}_{ij}$. The structure has the following subfields:

   `.subi`

   `.subj`

   `.subk`

   `.subl`

   `.val`

   See Section 8.1.5 for details on this structure.

`.buc`

   Upper bounds of the constraints. $\infty$ denotes an infinite upper bound. If the field is not
   defined or `buc==[]`, then all the upper bounds are assumed to be equal to $\infty$.

.blx

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

Upper bounds on the variables. $\infty$ denotes an infinite upper bound. If the field is not defined or `bux==[]`, then all the upper bounds are assumed to be equal to $\infty$.

.ints

A MATLAB structure which has the subfields

```
.sub; % Required.
.pri; % Subfields.
```

`ints.sub` is a one-dimensional array containing the indexes of the integer-constrained variables. Hence, `ints.sub` is identical to the set $\mathcal{J}$ in (8.2). `ints.pri` is also 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

A MATLAB structure defining the conic constraints (8.1). See Section 8.1.3 for details.

.sol

A MATLAB structure containing a guess on the optimal solution which some of the optimizers in MOSEK may exploit. See Section 8.1.6 for details on this structure.

.primalrepair

A MATLAB 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

A MATLAB structure which has the subfields:

.cons.subu

Indeces of constraints, where upper bounds are analyzed for sensitivity.

.cons.subl

Indeces of constraints, where lower bounds are analyzed for sensitivity.

.vars.subu

Indeces of variables, where upper bounds are analyzed for sensitivity.

.vars.subl

Indeces of variables, where lower bounds are analyzed for sensitivity.

.sub

Index of variables where coefficients are analysed for sensitivity.

Comments:

MOSEK solves linear, quadratic, quadratically constrained, and conic optimization problems. The simplest of those is a linear problem, which is posed in MOSEK as

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

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

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

where the conic constraint

$$x \in \mathcal{C} \tag{8.1}$$

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

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

The matrix

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

must be symmetric positive semidefinite and the matrix

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

must be either symmetric negative semidefinite with the $i$th constraint

$$l_i^c \leq \frac{1}{2}\sum_{j=1}^n \sum_{k=1}^n q_{j,k}^i x_j x_k + \sum_{j=1}^n a_{i,j} x_j,$$

or $Q^i$ must be symmetric positive semidefinite with the $i$th constraint

$$\frac{1}{2}\sum_{j=1}^n \sum_{k=1}^n q_{j,k}^i x_j x_k + \sum_{j=1}^n a_{i,j} x_j \leq u_i^c.$$

Note that the if the quadratic terms $Q^i$ are absent, the problem reduces to a standard quadratic optimization problem.

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

$$x_j \text{ integer-constrained for all } j \in \mathcal{J} \tag{8.2}$$

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

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

  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 $\text{tr}(\overline{C}_j\overline{X}_j)$ 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 semidefinite problems. The matrices are specifed in triplet format discard zero elements, and since they are symmetric only the lower triangulars should be specified.

- $l^c$ specifies the lower bounds of the constraints.

- $u^c$ specifies the upper bounds of the constraints.

- $l^x$ specifies the lower bounds on the variables $x$.

- $u^x$ specifies the upper bounds on the variables $x$.

- In conic problems, a partitioning of $x$ belongs to a set of free variables and quadratic cones. Let

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

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

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

Next, define

$$\mathcal{C} := \left\{ x \in \mathbb{R}^n : \ x^t \in \mathcal{C}_t, \ t = 1, \ldots, k \right\}$$

where $\mathcal{C}_t$ must have one of the following forms

- Free variables:

$$\mathcal{C}_t = \{ x \in \mathbb{R}^{n^t} \}.$$

- Quadratic cones:

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

- Rotated quadratic cones:

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

All the parameters of the optimization problem are stored using one or more subfields of the `prob` structure using the naming convention in Table 8.1. In Table 8.1 all the parameters are listed with their corresponding type. The `int` type indicates that the field must contain an integer value, `double` indicates a real number. The relationship between $Q^o$ and $Q^p$ and the subfields of the `prob` structure is as follows:

| Field name | Type | Dimension | Optional | Problem parameter |
|---|---|---|---|---|
| qosubi | int | length(qoval) | Yes | $q^o_{ij}$ |
| qosubj | int | length(qoval) | Yes | $q^o_{ij}$ |
| qoval | double | length(qoval) | Yes | $q^o_{ij}$ |
| c | double | $n$ | Yes | $c_j$ |
| qcsubk | int | length(qcval) | Yes | $q^p_{ij}$ |
| qcsubi | int | length(qcval) | Yes | $q^p_{ij}$ |
| qcsubj | int | length(qcval) | Yes | $q^p_{ij}$ |
| qcval | double | length(qcval) | Yes | $q^p_{ij}$ |
| a | Sparse matrix | $m \times n$ | No | $a_{ij}$ |
| bardim | int | $p$ | Yes | $r_j$ |
| barc | MATLAB structure | | Yes | $\overline{C}_j$ |
| bara | MATLAB structure | | Yes | $\overline{A}_{ij}$ |
| blc | double | $m$ | Yes | $l^c_k$ |
| buc | double | $m$ | Yes | $u^c_k$ |
| blx | double | $n$ | Yes | $l^x_k$ |
| bux | double | $n$ | Yes | $u^x_k$ |
| ints | MATLAB structure | $|\mathcal{J}|$ | Yes | $\mathcal{J}$ |
| cones | MATLAB cell array | $k$ | Yes | $\mathcal{C}$ |

Table 8.1: The relation between fields and problem parameters.

- The quadratic terms in the objective:

$$q^o_{\texttt{qosubi(t),qoval(t)}} = \texttt{qoval(t)}, \ t = 1, 2, \ldots, \texttt{length(qoval)}. \tag{8.3}$$

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

- The quadratic terms in the constraints:

$$q^{\texttt{qcsubk(t)}}_{\texttt{qcsubi(t),qcsubj(t)}} = \texttt{qcval(t)}, \ t = 1, 2, \ldots, \texttt{length(qcval)}. \tag{8.4}$$

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

## 8.1.2 names

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

.name

Contains the problem name.

**.obj**

> Contains the name of the objective.

**.con**

> Is a MATLAB cell array where `names.con{i}` contains the name of the $i$th constraint.

**.var**

> Is a MATLAB cell array where `names.var{j}` contains the name of the $j$th variable.

**.barvar**

> Is a MATLAB cell array where `names.barvar{j}` contains the name of the $j$th semidefinite variable.

**.cone**

> Is a MATLAB cell array where `names.cone{t}` contains the name of the $t$th conic constraint.

### 8.1.3   `cones`

`cones` is a MATLAB structure with the following subfields:

**.type**

> `cones.type` is 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**

> `cones.sub` is an array of variable indexes specifying which variables are members of the cones. The array is a concatenation of index lists of all the cones.

**.subptr**

> `cones.sub` is an array of pointers into `cones.sub` indicating the beginning of the different cone index-sets.

For example the quadratic cone

$$x_5 \geq \sqrt{x_3^2 + x_1^2}$$

and rotated quadratic cone

$$2x_6 x_4 \geq x_2^2 + x_7^2$$

would be specified using the two arrays

```
cones.type   = [0, 1];
cones.sub    = [5, 3, 1, 6, 4, 2, 7];
cones.subptr = [1, 4];
```

Alternatively, `cones` can be specified as a cell array with the sub-fields

`.type`

> `cones{i}.type` contains the cone type for the $i$th cone; 'MSK_CT_QUAD' or MSK_CT_RQUAD, indicating if cone is a quadratic cone or a rotated quadratic cone.

`.sub`

> `cones{i}.sub` is a list of variable indexes specifying which variables are members of the cone.

This specification is deprecated and might be removed in future versions of MOSEK.

### 8.1.4  `barc`

Together with `.bardim` this structure specifies the symmetric matrices $\overline{C}_j$ in the objective for semidefinite problems.

The structure has the following fields, which must be arrays of the same length:

`.subj`

> Semidefinite variable indices $j$.

`.subk`

> Subscripts of nonzeros elements.

`.subl`

> Subscripts of nonzeros elements.

`.val`

> Numerical values.

The symmetric matrices are specified in block-triplet format as

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

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

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

and that

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

$$1 \leq \texttt{barc.subl(t)} \leq \texttt{bardim(barc.subj(t))}, \ t = 1, 2, \ldots, \texttt{length(barc.subj)}.$$

## 8.1.5   bara

Together with `.bardim` this structure specifies the symmetric matrices $\overline{A}_{ij}$ in the constraints of semidefinite problems.

The structure has the following fields, which must be arrays of the same length:

`.subi`

> Constraint indices $i$.

`.subj`

> Semidefinite variable indices $j$.

`.subk`

> Subscripts of nonzeros elements.

`.subl`

> Subscripts of nonzeros elements.

`.val`

> Numerical values.

The symmetric matrices are specified in block-triplet format as

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

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

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

and that

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

$$1 \leq \texttt{bara.subl(t)} \leq \texttt{bardim(bara.subj(t))}, \ t = 1, 2, \ldots, \texttt{length(bara.subj)}.$$

## 8.1.6   sol

Description:

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

Subfields:

.itr

Interior (point) solution computed by the interior-point optimizer.

.bas

Basic solution computed by the simplex optimizers and basis identification procedure.

.int

Integer solution computed by the mixed-integer optimizer.

Comments:

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

.prosta

Problem status. See Appendix MSKprostae.

.solsta

Solution status. See Appendix MSKsolstae.

.skc

Constraint status keys. See Table 7.1.

.skx

Variable status keys. See Table 7.1.

.skn

Conic status keys. See Section 7.1.

.xc

Constraint activities, i.e., $x_c = Ax$ where $x$ is the optimal solution.

.xx

The optimal $x$ solution.

.barx

The optimal solution of $\overline{X}_j, \ j = 1, 2, \ldots, \texttt{length(bardim)}$.

.bars

The optimal solution of $\overline{S}_j, \ j = 1, 2, \ldots, \texttt{length(bardim)}$.

.y

Identical to `sol.slc-sol.suc`.

.slc

Dual solution corresponding to the lower constraint bo-unds.

.suc

Dual solution corresponding to the upper constraint bo-unds.

.slx

Dual solution corresponding to the lower variable bou-nds.

.sux

Dual solution corresponding to the upper variable bou-nds.

.snx
> Dual solution corresponding to the conic constraint.

.pobjval
> The primal objective value.

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

### 8.1.7  prisen

Description:
> Results of the primal sensitivity analysis.

Subfields:

.cons
> MATLAB structure with the subfields:
>
> .lr_bl
> > Left value $\beta_1$ in the linearity interval for a lower bound.
>
> .rr_bl
> > Right value $\beta_2$ in the linearity interval for a lower bound.
>
> .ls_bl
> > Left shadow price $s_l$ for a lower bound.
>
> .rs_bl
> > Right shadow price $s_r$ for a lower bound.
>
> .lr_bu
> > Left value $\beta_1$ in the linearity interval for an upper bound.
>
> .rr_bu
> > Right value $\beta_2$ in the linearity interval for an upper bound.
>
> .ls_bu
> > Left shadow price $s_l$ for an upper bound.
>
> .rs_bu
> > Right shadow price $s_r$ for an upper bound.

.var
> MATLAB structure with the subfields:
>
> .lr_bl
> > Left value $\beta_1$ in the linearity interval for a lower bound on a variable.
>
> .rr_bl
> > Right value $\beta_2$ in the linearity interval for a lower bound on a variable.
>
> .ls_bl
> > Left shadow price $s_l$ for a lower bound on a variable.

.rs_bl
    Right shadow price $s_r$ for a lower bound on a variable.

.lr_bu
    Left value $\beta_1$ in the linearity interval for an upper bound on a variable.

.rr_bu
    Right value $\beta_2$ in the linearity interval for an upper bound on a variable.

.ls_bu
    Left shadow price $s_l$ for an upper bound on a variable.

.rs_bu
    Right shadow price $s_r$ for an upper bound on a variable.

### 8.1.8   duasen

Description:

    Results of dual the sensitivity analysis.

Subfields:

.lr_c
    Left value $\beta_1$ in linearity interval for an objective coefficient.

.rr_c
    Right value $\beta_2$ in linearity interval for an objective coefficient.

.ls_c
    Left shadow price $s_l$ for an objective coefficient.

.rs_c
    Right shadow price $s_r$ for an objective coefficient.

### 8.1.9   info

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

### 8.1.10   symbcon

symbcon is a MATLAB structure containing a subfield for each MOSEK symbolic constant, e.g., the field symbcon.MSK_DINF_BI_TIME specifies the value of the symbolic constant "MSK_DINF_BI_TIME". In Appendix 25 allthe symbolic constants are listed.

### 8.1.11   `callback`

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

`.loghandle`

>   A MATLAB data structure or just `[]`.

`.log`

>   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`

>   A MATLAB data structure or just `[]`.

`.iter`

>   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 8.1.9 for further details about the `info` data structure.

## 8.2   An example of a command reference

All functions are documented using the format:

`somefunction`

>   Description:
>   >   The purpose of the function.
>
>   Syntax:
>   >   `[ret1,ret2] = somefunction(arg1,arg2)`
>
>   Arguments:
>   >   `arg1`
>   >   >   A description of this argument.
>   >
>   >   `arg2`
>   >   >   (Optional) A description of this argument which is optional. However, if argument 3 is specified, this argument must be specified too.
>   >
>   >   `arg3`
>   >   >   Another useful argument.
>
>   Returns:

`ret1`
>   A description of the first return `ret1`.

`ret2`
>   (Optional) A description of the second return `ret2`.

Comments:

>   Potentially some comments about the function.

Examples:

>   Some examples of the use of the function.

## 8.3 Functions provided by the MOSEK optimization toolbox

`mosekopt`

Description

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

Syntax:

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

Arguments:

`cmd`

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

>   `anapro`
>>   Runs the problem analyzer.

>   `echo(n)`
>>   Controls how much information is echoed to the scre-en. Here, `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 `res.info`. This database contains various task specific information. See Section 8.1.9 for details the `info` data structure.

>   `param`
>>   Return the complete parameter database in `res.param`.

>   `primalrepair`
>>   Performs a primal feasibility repair. See Chapter 15 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, where `statuskeys(0)` means that all the status keys are returned as strings, and `statuskeys(1)` means that all the status keys are returned as numeric codes.

symbcon

    Return the `symbcon` data structure in `res.symbcon`. See Section 8.1.10 for details on the `symbcon` data structure.

write(name)

    Write problem to the file "`name`".

version

    Return the MOSEK version numbers in `res.version`.

prob

    (Optional) A MATLAB structure containing the problem data. See Table 8.1 for details.

param

    (Optional) A MATLAB structure which is used to specify algorithmic parameters to MOSEK. The fields of `param` must be valid MOSEK parameter names. Moreover, the values corresponding to the fields must be of a valid type, i.e. the value of a string parameter must be a string, the value of an integer paramter must be an integer etc.

callback

    (Optional) A MATLAB structure defining call-back data and functions. See Sections 7.24 and 8.1.11 for details.

Returns:

rcode

    Return code. The interpretation of the value of the return code is listed in Appendix 25.

res

    (Optional) Solution obtained by the interior-point algorithm.

    .sol

        The data structure of the type `sol` is discussed in Section 8.1.

.info
> A MATLAB structure containing the task information database which contains various task related information such as the number of iterations used to solve the problem. However, this field is only defined if `info` appeared in the `cmd` command when `mosekopt` is invoked.

.param
> A MATLAB structure which contain the complete MOSEK parameter database. However, this field is defined only if the `param` command is present in `cmd` when `mosekopt` is invoked.

.prob
> Contains the problem data if the problem data was read from a file.

Examples:
> The following example demonstrates how to display the MOSEK parameter database and how to use the primal simplex optimizer instead of the default optimizer.

## msklpopt

## mskqpopt

## mskenopt

## mskgpopt

## mskscopt

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

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

Arguments:
> For a description of the arguments we refer the reader to the actual m files stored in
> ```
> <root>\mosek\7\toolbox\matlab\solvers
> ```
> Please note that the MATLAB command `help`,
> ```
> help msklpopt
> ```
> will produce some usage information for the functions.

Returns:
> Identical to the `res` structure returned by `mosekopt`

## mskgpwri

Description

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

Syntax:

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

Arguments:

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

`filename`
The output file name.

Returns:

Nothing.

### mskgpread

Description

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

Syntax:

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

Arguments:

`filename`
The name of the file to read.

Returns:

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

## 8.4   MATLAB optimization toolbox compatible functions

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

### 8.4.1   Linear and quadratic optimization

#### linprog

Description

Solves the linear optimization problem:

$$\begin{array}{lrcl} \text{minimize} & f^T x & & \\ \text{subject to} & Ax & \leq & b, \\ & Bx & = & c, \\ & \multicolumn{3}{l}{l \leq x \leq u.} \end{array}$$

Syntax:

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

Syntax:

```
[x,fval,exitflag,output,lambda]
  = linprog(problem)
```

Arguments:

f

The objective function.

A

Constraint matrix for the less-than equal inequalities. Use $A = []$ if there are no inequalities.

b

Right-hand side for the less-than equal inequalities. Use $b = []$ if there are no inequalities.

B

(Optional) Constraint matrix for the equalities. Use $B = []$ if there are no equalities.

c

(Optional) Right-hand side for the equalities. Use $c = []$ if there are no equalities.

l

(Optional) Lower bounds on the variables. Please use $-\infty$ to represent infinite lower bounds.

u

(Optional) Upper bounds on the variables. Please use $\infty$ to represent infinite upper bounds.

x0

(Optional) An initial guess for the starting point. This is only used for the primal simplex algorithm; For more advanced hot-starting (e.g., using dual simplex), use `mosekopt` directly.

options

(Optional) An optimization options structure. See the `mskoptimset` function for the definition of the optimization options structure. `linprog` uses the options

.Diagnostics

.Display

.MaxIter

.Simplex

Valid values are 'on', 'primal' or 'dual'. If `Simplex` is 'on' then MOSEK will use either a primal or dual simplex solver (similar as specifying 'MSK_OPTIMIZER_FREE_SIMPLEX'

in `mosekopt`; otherwise either a primal or dual simplex algorithm is used. Note, that the `'primal'` and `'dual'` values are specific for the MOSEK interface, and not present in the standard MATLAB version.

.**Write**
Filename of problem file (e.g., `'prob.opf'`) to be saved. This is useful for reporting bugs or problems.

problem
A structure containing the fields $f$, $A$, $b$, $B$, $c$, $l$, $u$, $x0$ and *options*.

Returns:

x
The optimal $x$ solution.

fval
The optimal objective value, i.e. $f^T x$.

exitflag
A number which has the interpretation:

$< 0$
The problem is likely to be either primal or dual infeasible.

$= 0$
The maximum number of iterations was reached.

$> 0$
$x$ is an optimal solution.

output

.**iterations**
Number of iterations spent to reach the optimum.

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

lambda

.**lower**
Lagrange multipliers for lower bounds $l$.

.**upper**
Lagrange multipliers for upper bounds $u$.

.**ineqlin**
Lagrange multipliers for the inequalities.

.**eqlin**
Lagrange multipliers for the equalities.

**quadprog**

Description
Solves the quadratic optimization problem:

$$
\begin{aligned}
\text{minimize} \quad & \frac{1}{2} x^T H x + f^T x \\
\text{subject to} \quad & Ax \quad \leq \quad b, \\
& Bx \quad = \quad c, \\
& l \leq x \leq u.
\end{aligned}
\tag{8.5}
$$

Syntax:

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

Arguments:

H

Hessian of the objective function. $H$ must be a symmetric matrix. Contrary to the MATLAB optimization toolbox, MOSEK handles only the cases where $H$ is positive semidefinite. On the other hand MOSEK always computes a global optimum, i.e. the objective function has to be strictly convex.

f

See (8.5) for the definition.

A

Constraint matrix for the less-than equal inequalities. Use $A = []$ if there are no inequalities.

b

Right-hand side for the less-than equal inequalities. Use $b = []$ if there are no inequalities.

B

(Optional) Constraint matrix for the equalities. Use $B = []$ if there are no equalities.

c

(Optional) Right-hand side for the equalities. Use $c = []$ if there are no equalities.

l

(Optional) Lower bounds on the variables. Please use $-\infty$ to represent infinite lower bounds.

u

(Optional) Upper bounds on the variables. Please use $\infty$ to represent infinite upper bounds.

x0

(Optional) An initial guess for the starting point. This information is ignored by MOSEK

options

(Optional) An optimization options structure. See the `mskoptimset` function for the definition of the optimizations options structure. `qu-ad-prog` uses the options

.Diagnostics

.Display

.MaxIter

.Write

Returns:

x

The $x$ solution.

fval

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

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

Examples:

## 8.4.2   Linear optimization with binary variables

bintprog

Description
    Solves the binary linear optimization problem:

$$
\begin{array}{ll}
\text{minimize} & f^T x \\
\text{subject to} & Ax \leq b, \\
& Bx = c, \\
& x \in \{0,1\}^n
\end{array}
$$

Syntax:
    [x,fval,exitflag,output] = bintprog(f,A,b,B,c,x0,options)
Syntax:
    [x,fval,exitflag,output] = bintprog(problem)
Arguments:

    f
        The objective function.

A

Constraint matrix for the less-than equal inequalities. Use $A = []$ if there are no inequalities.

b

Right-hand side for the less-than equal inequalities. Use $b = []$ if there are no inequalities.

B

(Optional) Constraint matrix for the equalities. Use $B = []$ if there are no equalities.

c

(Optional) Right-hand side for the equalities. Use $c = []$ if there are no equalities.

x0

(Optional) A feasible starting point.

options

(Optional) An optimization options structure. See the `mskoptimset` function for the definition of the optimization options structure. `bintprog` uses the options

`.Diagnostics`

`.Display`

`.MaxTime`

The maximum number of seconds in the solution-time

`.MaxNodes`

The maximum number of branch-and-bounds allowed

`.Write`

Filename of problem file to save.

problem

A structure containing the fields $f$, $A$, $b$, $B$, $c$, $x0$ and *options*.

Returns:

x

The solution $x$.

fval

The objective $f^T x$.

exitflag

A number which has the interpretation:

1

The function returned an integer feasible solution.

$-2$

The problem is infeasible.

$-4$

`maxNodes` reached without converging.

$-5$

`maxTime` reached without converging.

## 8.4.3  For linear least squares problems

lsqlin

Description

Solves the linear least squares problem:

$$
\begin{array}{lrcl}
\text{minimize} & \dfrac{1}{2}\,\|Cx - d\|_2^2 \\
\text{subject to} & Ax & \leq & b, \\
& Bx & = & c, \\
& l \leq x \leq u.
\end{array}
\tag{8.6}
$$

Syntax:

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

Arguments:

C

A matrix. See problem (8.6) for the purpose of the argument.

d

A vector. See problem (8.6) for the purpose of the argument.

A

Constraint matrix for the less-than equal inequalities. Use $A = []$ if there are no inequalities.

b

Right-hand side for the less-than equal inequalities. Use $b = []$ if there are no inequalities.

B

(Optional) Constraint matrix for the equalities. Use $B = []$ if there are no equalities.

c

(Optional) Right-hand side for the equalities. Use $c = []$ if there are no equalities.

l

(Optional) Lower bounds on the variables. Please use $-\infty$ to represent infinite lower bounds.

u

(Optional) Upper bounds on the variables. Please use $\infty$ to represent infinite lower bounds.

x0

(Optional) An initial guess for the starting point. This information is ignored by MOSEK

options

(Optional) An optimization options structure. See the function `mskoptimset` function for the definition of the optimization options structure. `lsqprog` uses the options

`.Diagnostics`

`.Display`

`.MaxIter`

`.Write`

Returns:

x
>   The optimal $x$ solution.

resnorm
>   The squared norm of the optimal residuals, i.e. $\|Cx - d\|^2$ evaluated at the optimal solution.

residual
>   The residual $Cx - d$.

exitflag
>   A scalar which has the interpretation:
>
>   $< 0$
>   >   The problem is likely to be either primal or dual infeasible.
>
>   $= 0$
>   >   The maximum number of iterations was reached.
>
>   $> 0$
>   >   $x$ is the optimal solution.

output
>   .iterations
>   >   Number of iterations spent to reach the optimum.
>
>   .algorithm
>   >   Always defined as 'large-scale: interior-point'.

lambda
>   .lower
>   >   Lagrange multipliers for lower bounds $l$.
>
>   .upper
>   >   Lagrange multipliers for upper bounds $u$.
>
>   .ineqlin
>   >   Lagrange multipliers for inequalities.
>
>   .eqlin
>   >   Lagrange multipliers for equalities.

Examples:

## lsqnonneg

Description
>   Solves the linear least squares problem:

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2} \|Cx - d\|_2^2 \\
\text{subject to} \quad & x \geq 0.
\end{aligned} \tag{8.7}$$

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

Arguments:

C

    See problem (8.7).

d

    See problem (8.7).

x0

    (Optional) An initial guess for the starting point. This information is ignored by MOSEK

options

    (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`

Returns:

x

    The $x$ solution.

resnorm

    The squared norm of the optimal residuals, i.e.

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

    evaluated at the optimal solution.

residual

    The residual $Cx - d$.

exitflag

    A number which has the interpretation:

    $< 0$

      The problem is likely to be either primal or dual infeasible.

    $= 0$

      The maximum number of iterations was reached.

    $> 0$

      $x$ is optimal solution.

output

    `.iterations`

      Number of iterations spend to reach the optimum.

    `.algorithm`

      Always defined to be 'large-scale: interior-point'.

lambda

    `.lower`

      Lagrange multipliers for lower bounds $l$.

    `.upper`

      Lagrange multipliers for upper bounds $u$.

`.ineqlin`
    Lagrange multipliers for inequalities.

`.eqlin`
    Lagrange multipliers for equalities.

Comments:

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

Examples:

## 8.4.4 The optimization options

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

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

### 8.4.4.1 Viewing and modifying the optimization options

`mskoptimget`

Description

Obtains a value of an optimization parameter.

Syntax:

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

Arguments:

`options`
    The optimization options structure.

`param`
    Name of the optimization parameter for which the value should be obtained.

`default`
    (Optional) If `param` is not defined, the value of `default` is returned instead.

Returns:

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

Comments:

See the `mskoptimset` function for which parameters that can be set.

Examples:

`mskoptimset`

Description

Obtains and modifies the optimization options structure. Only a subset of the fields in the optimization structure recognized by the MATLAB optimization toolbox is recognized by MOSEK In addition the optimization options structure can be used to modify all the MOSEK specific parameters defined in Appendix 23.

.`Diagnostics`

Used to control how much diagnostic information is printed. Following values are accepted:

`off`

No diagnostic information is printed.

`on`

Diagnostic information is printed.

.`Display`

Defines what information is displayed. The following values are accepted:

`off`

No output is displayed.

`iter`

Some output is displayed for each iteration.

`final`

Only the final output is displayed.

.`MaxIter`

Maximum number of iterations allowed.

.`Write`

A filename to write the problem to. If equal to the empty string no file is written. E.g the option
`Write(myfile.opf)`
writes the file `myfile.opf` in the `opf` format.

Syntax:

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

Arguments:

`arg1`

(Optional) Is allowed to be any of the following two things:

Any string

The same as using no argument.

A structure

The argument is assumed to be a structure containing options, which are copied to the return options.

`param1`

(Optional) A string containing the name of a parameter that should be modified.

value1
  (Optional) The new value assigned to the parameter with the name `param1`.

param2
  (Optional) Has the same interpretation as `param1`.

value2
  (Optional) Has the same interpretation as `value1`.

Returns:

options
  The updated optimization options structure.

Examples:

# Chapter 9

# Case studies

## 9.1 Robust linear optimization

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

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

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

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

### 9.1.1 Introductory example

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

|  | DrugI | DrugII |
|---|---|---|
| Selling price $ per 1000 packs | 6200 | 6900 |
| Contentof 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, $ per kg | Content of agent A, gm per kg |
|---|---|---|
| RawI | 100.00 | 0.01 |
| RawII | 199.90 | 0.02 |

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

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

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

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

and maximize the income

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

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

Minimize

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

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 (9.1) denotes the profit to be maximized, and the inequalities can be interpreted as follows:

- Balance of the active agent.

- Storage restriction.

- Manpower restriction.

- Equipment restriction.

- Ducget restriction.

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

──────────────────────────────────────────────[ rlo1.m ]──────────────────────────────────────────────

```
1   % rlo1.m
2
3   clear prob;
4
5   prob.c   = [-100;-199.9;6200-700;6900-800];
6   prob.a   = sparse([0.01,0.02,-0.500,-0.600;1,1,0,0;
7                      0,0,90.0,100.0;0,0,40.0,50.0;100.0,199.9,700,800]);
8   prob.blc = [0;-inf;-inf;-inf;-inf];
9   prob.buc = [inf;1000;2000;800;100000];
10  prob.blx = [0;0;0;0];
11  prob.bux = [inf;inf;inf;inf];
12  [r,res]  = mosekopt('maximize',prob);
13  xx       = res.sol.itr.xx;
14  RawI     = xx(1);
15  RawII    = xx(2);
16  DrugI    = xx(3);
17  DrugII   = xx(4);
18
19  disp(sprintf('*** Optimal value: %8.3f',prob.c'*xx));
20  disp('*** Optimal solution:');
21  disp(sprintf('RawI:   %8.3f',RawI));
22  disp(sprintf('RawII:  %8.3f',RawII));
23  disp(sprintf('DrugI:  %8.3f',DrugI));
24  disp(sprintf('DrugII: %8.3f',DrugII));
```

────────────────────────────────────────────────────────────────────────────────────────────────────

When executing this script, the following is displayed:

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

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

## 9.1.2 Data uncertainty and its consequences.

Please note that not all problem data can be regarded as "absolutely reliable"; e.g. one can hardly believe that the contents of the active agent in the raw materials are *exactly* the "nominal data" 0.01 gm/kg for RawI and 0.02 gm/kg for RawII. In reality, these contents definitely vary around the indicated values. A natural assumption here is that the actual contents of the active agent $a_I$ in RawI and $a_{II}$ in RawII are realizations of random variables somehow distributed around the "nominal contents" $a_I^n = 0.01$ and $a_{II}^n = 0.02$. To be more specific, assume that $a_I$ drifts in the 0.5% margin of

$a_I^n$, i.e. it takes with probability 0.5 the values from the interval $a_I^n(1\pm0.005) = a_I^n\{0.00995; 0.01005\}$. Similarly, assume that $a_{II}$ drifts in the 2% margin of $a_{II}^n$, taking with probabilities 0.5 the values $a_{II}^n(1\pm0.02) = a_I^n\{0.0196; 0.0204\}$. 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 reported in [6] demonstrates that for 13 of 94 NETLIB problems, already 0.01% perturbations of "clearly uncertain" data can make the nominal optimal solution severely infeasible: with these perturbations, the solution, with a non-negligible probability, violates some of the constraints by 50% and more. It should be added that in the general case, in contrast to the toy example we have considered, there is no evident way to adjust the optimal solution by a small modification to the actual values of the data. Moreover there are cases when such an adjustment is impossible — in order to become feasible for the perturbed data, the nominal optimal solution should be "completely reshaped".

### 9.1.3 Robust linear optimization methodology

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

#### 9.1.3.1 Uncertain linear programs and their robust counterparts.

Consider a linear optimization problem

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

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

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

Here

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

is the *nominal data,*

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

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

- The family of instances (9.2) with data running through a given uncertainty set $\mathcal{U}$ is called an *uncertain linear optimization problem.*

- Vector $x$ is called a *robust feasible solution* to an uncertain linear optimization problem, if it remains feasible for all realizations of the data from the uncertainty set, i.e. if

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

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

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

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

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

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

In general, the RC (9.4) of an uncertain linear optimization problem is not a linear optimization problem since(9.4) has infinitely many linear constraints. There are, however, cases when (9.4) can

be rewritten equivalently as a linear programming program; in particular, this is the case for interval uncertainty (9.3). Specifically, in the case of (9.3), the robust counterpart of uncertain linear program is equivalent to the following linear program in variables $x, y, t$:

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

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

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

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

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

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

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

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

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

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

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

and that

$$y = g + x$$

Substituting these values into (9.5) gives

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

which after some simplifications leads to

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

and

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

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

### 9.1.3.3  Introductory example (continued)

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

$$\text{(Drug\_RC)} :$$
$$\text{maximize}$$
$$t$$
$$\text{subject to}$$

$$
\begin{array}{rcl}
t \leq -100 \cdot \mathtt{RawI} - 199.9 \cdot \mathtt{RawII} + 5500 \cdot \mathtt{DrugI} + 6100 \cdot \mathtt{DrugII} & & \\
0.01 \cdot 0.995 \cdot \mathtt{RawI} + 0.02 \cdot 0.98 \cdot \mathtt{RawII} - 0.500 \cdot \mathtt{DrugI} - 0.600 \cdot \mathtt{DrugII} & \geq & 0 \\
\mathtt{RawI} + \mathtt{RawII} & \leq & 1000 \\
90.0 \cdot \mathtt{DrugI} + 100.0 \cdot \mathtt{DrugII} & \leq & 2000 \\
40.0 \cdot \mathtt{DrugI} + 50.0 \cdot \mathtt{DrugII} & \leq & 800 \\
100.0 \cdot \mathtt{RawI} + 199.90 \cdot \mathtt{RawII} + 700 \cdot \mathtt{DrugI} + 800 \cdot \mathtt{DrugII} & \leq & 100000 \\
\mathtt{RawI}, \mathtt{RawII}, \mathtt{DrugI}, \mathtt{DrugII} & \geq & 0
\end{array}
$$

Solving this problem with MOSEK we get the following output:

```
*** Optimal value: 8294.567
*** Optimal solution:
RawI:    877.732
RawII:     0.000
DrugI:    17.467
DrugII:    0.000
```

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

### 9.1.4   Random uncertainty and ellipsoidal robust counterpart

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

$$a^T x \leq b \tag{9.7}$$

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

$$a = a^{\mathrm{n}} + \zeta, \tag{9.8}$$

where $a^{\mathrm{n}}$ is the vector of nominal coefficients and $\zeta$ is a random perturbation vector with zero mean and covariance matrix $V_a$. In this case the value of the left-hand side of (9.7), evaluated at a given $x$, becomes a random variable with the expected value $(a^{\mathrm{n}})^T x$ and the standard deviation $\sqrt{x^T V_a x}$. Now let us act as an engineer who believes that the value of a random variable never exceeds its mean plus 3 times the standard deviation; we do not intend to be that specific and replace "3" in the above rule by a safety parameter $\Omega$ which will be in our control. Believing that the value of a random variable "never" exceeds its mean plus $\Omega$ times the standard deviation, we conclude that a "safe" version of (9.7) is the inequality

$$(a^{\mathrm{n}})^T x + \Omega \sqrt{x^T V_a x} \le b. \tag{9.9}$$

The word "safe" above admits a quantitative interpretation: If $x$ satisfies (9.9), one can bound from above the probability of the event that random perturbations (9.8) result in violating the constraint (9.7) evaluated at $x$. The bound in question depends on what we know about the distribution of $\zeta$, e.g.

- We always have the bound given by the Tschebyshev inequality: $x$ satisfies (9.9)$\Rightarrow$

$$\mathrm{Prob}\left\{a^T x > b\right\} \le \frac{1}{\Omega^2}. \tag{9.10}$$

- When $\zeta$ is Gaussian, then the Tschebyshev bound can be improved to: $x$ satisfies (9.9)$\Rightarrow$

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

- Assume that $\zeta = D\xi$, where $\Delta$ is certain $n \times m$ matrix, and $\xi = (\xi_1, ..., \xi_m)^T$ is a random vector with independent coordinates $\xi_1, ..., \xi_m$ symmetrically distributed in the segment $[-1, 1]$. Setting $V = DD^T$ ($V$ is a natural "upper bound" on the covariance matrix of $\zeta$), one has: $x$ satisfies (9.9)$\Rightarrow$

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

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

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

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

where $(c^{\mathrm{n}}, \{a_i^{\mathrm{n}}\}_{i=1}^m, l_c, u_c, l_x, u_x)$ are the nominal data, and $dc$, $da_i$ are random perturbations with zero means. Assume, for the sake of definiteness, that every one of the random perturbations $dc, da_1, ..., da_m$ satisfies either the assumption of item 2 or the assumption of item 3, and let $V_c, V_1, ..., V_m$ be the

corresponding (upper bounds on the) covariance matrices of the perturbations. Choosing a safety parameter $\Omega$ and replacing the objective and the bodies of all the constraints by their safe bounds as explained above, we arrive at the following optimization problem:

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

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

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

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

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

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

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

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

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

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

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

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

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

### 9.1.4.1 Example: Interval and Ellipsoidal robust counterparts of uncertain linear constraint with independent random perturbations of coefficients

Consider a linear constraint

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

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

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

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

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

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

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

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

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

For a moderate value of $\Omega$, say $\Omega = 6$, and $n \geq 40$, the ellipsoid $E(\Omega)$ in its diameter, typical linear sizes, volume, etc. is incomparably less than the box $B$, the difference becoming more dramatic the larger the dimension $n$ of the box and the ellipsoid. It follows that the ellipsoidal robust counterpart (9.17) of the randomly perturbed uncertain constraint (9.15) is much less conservative than the interval robust counterpart (9.16), while ensuring basically the same "robustness guarantees". To illustrate this important point, consider the following numerical examples:

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

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

$$
\begin{array}{rll}
\text{maximize} & \displaystyle\sum_{j=1}^{n} a_j x_j & \\[4mm]
\text{subject to} & \displaystyle\sum_{j=1}^{n} x_j & = \quad 1, \\[4mm]
0 \quad \leq & x_j, & j = 1, ..., n.
\end{array}
\tag{9.18}
$$

where $a_j$ are the uncertain returns of the assets. Both the nominal optimal solution (set all returns $a_j$ equal to their nominal values $\delta_j$) and the risk-neutral Stochastic Programming approach (maximize the expected total return) result in the same solution: Our \$ 1 should be invested in the most promising asset(s) – the one(s) with the maximal nominal return. This solution, however, can be very unreliable if, as is typically the case in reality, the most promising asset has the largest volatility $\sigma$ and is in this sense the most risky. To reduce the risk, one can use the Robust Counterpart approach which results in the following optimization problems.

**The Interval Model of Uncertainty:**

$$
\begin{array}{rll}
\text{maximize} & t & \\[4mm]
\text{subject to} & 0 \quad \leq \quad -t + \displaystyle\sum_{j=1}^{n}(\delta_j - \sigma_j)x_j, & \\[4mm]
& \displaystyle\sum_{j=1}^{n} x_j \quad = \quad 1, & \\[4mm]
& 0 \quad \leq \quad x_j, \qquad j = 1, ..., n &
\end{array}
\tag{9.19}
$$

and

**The ellipsoidal Model of Uncertainty:** maximize $t$

subject to

$$0 \leq -t + \sum_{j=1}^{n}(\delta_j)x_j - \Omega z,$$

$$0 \leq z - \sqrt{\sum_{j=1}^{n}\sigma_j^2 x_j^2},$$

$$\sum_{j=1}^{n}x_j = 1,$$

$$0 \leq x_j, \qquad j = 1, ..., n.$$

$$(9.20)$$

Note that the problem (9.20) is essentially the risk-averted portfolio model proposed in mid-50's by Markowitz.

The solution of (9.19) is evident — our $ 1 should be invested in the asset(s) with the largest possible *guaranteed* return $\delta_j - \sigma_j$. In contrast to this very conservative policy (which in reality prescribes to keep the initial capital in a bank or in the most reliable, and thus low profit, assets), the optimal solution to (9.20) prescribes a quite reasonable diversification of investments which allows to get much better total return than (9.19) with basically zero risk. To illustrate this, assume that there are $n = 300$ assets with the nominal returns (per year) varying from 1.04 (bank savings) to 2.00:

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

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

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

Here is a MATLAB script which builds the associated problem (9.20), 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):

──────────────────────────────[ rlo2.m ]──────────────────────────────

```
1   % File: rlo2.m
2
3   % Problem:
4   %
5   % Maximize t subject to
6   % t <= sum(delta(j)*x(j)) -Omega*z,
7   % y(j) = sigma(j)*x(j), j=1,...,n,
8   % sum(x(j)) = 1,
9   % norm(y) <= z,
10  % 0 <= x.
11
12  clear prob;
13  n     = 300;
14  Omega = 6;
```

```
15
16  % Set nominal returns and volatilities
17  delta = (0.96/(n-1))*[0:1:n-1]+1.04;
18  sigma = (1.152/(n-1))*[0:1:n-1];
19
20  % Set mosekopt description of the problem
21  prob.c = -[1;zeros(2*n+1,1)];
22  A       = [-1,ones(1,n)+delta,-Omega,zeros(1,n);zeros(n+1,2*n+2)];
23  for j=1:n,
24      % Body of the constraint y(j) - sigma(j)*x(j) = 0:
25      A(j+1,j+1)   = -sigma(j);
26      A(j+1,2+n+j) = 1;
27  end;
28  A(n+2,2:n+1)          = ones(1,n);
29  prob.a               = sparse(A);
30  prob.blc             = [zeros(n+1,1);1];
31  prob.buc             = [inf;zeros(n,1);1];
32  prob.blx             = [-inf;zeros(n,1);0;zeros(n,1)];
33  prob.bux             = inf*ones(2*n+2,1);
34  prob.cones           = cell(1,1);
35  prob.cones{1}.type = 'MSK_CT_QUAD';
36  prob.cones{1}.sub  = [n+2;[n+3:1:2*n+2]'];
37
38  % Run mosekopt
39  [r,res]=mosekopt('minimize echo(1)',prob);
40
41  % Display the solution
42  xx = res.sol.itr.xx;
43  t  = xx(1);
44
45  disp(sprintf('Robust optimal value: %5.4f',t));
46  x = max(xx(2:1+n),zeros(n,1));
47  plot([1:1:n],x,'-m');
48  grid on;
49
50  disp('Press <Enter> to run simulations');
51  pause
52
53  % Run simulations
54
55  Nsim = 10000;
56  out  = zeros(Nsim,1);
57  for i=1:Nsim,
58      returns  = delta+(2*rand(1,n)-1).*sigma;
59      out(i)   = returns*x;
60  end;
61  disp(sprintf('Actual returns over %d simulations:',Nsim));
62  disp(sprintf('Min=%5.4f Mean=%5.4f Max=%5.4f StD=%5.2f',...
63      min(out),mean(out),max(out),std(out)));
64  hist(out);
```

Here are the results displayed by the script:

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

Please note that with our set-up there is exactly one asset with guaranteed return greater than 1 –
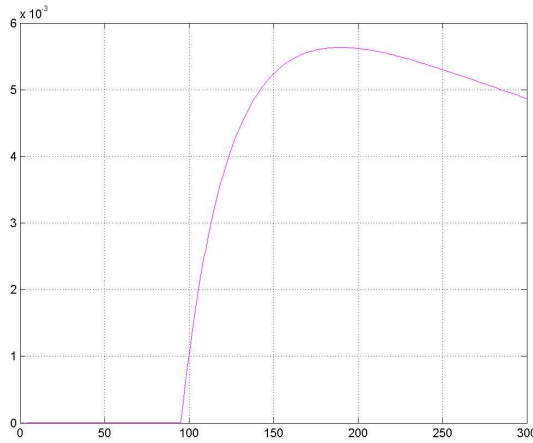
Figure 9.1: Distribution of investments among the assets in the optimal solution of.

asset # 1 (bank savings, return 1.04, zero volatility). Consequently, the interval robust counterpart (9.19) prescribes to put our $ 1 in the bank, thus getting a 4% profit. In contrast to this, the diversified portfolio given by the optimal solution of (9.20) never yields profit less than 57.2%, and yields at average a 69.67% profit with pretty low (0.03) standard deviation. We see that in favorable circumstances the ellipsoidal robust counterpart of an uncertain linear program indeed is less conservative than, although basically as reliable as, the interval robust counterpart.

Finally, let us compare our results with those given by the nominal optimal solution. The latter prescribes to invest everything we have in the most promising asset (in our example this is the asset # 300 with a nominal return of 2.00 and volatility of1.152). Assuming that the actual return is uniformly distributed in the corresponding interval and running 10,000 simulations, we get the following results:

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

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

### 9.1.4.2   Combined Interval-Ellipsoidal Robust Counterpart

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

- The interval uncertainty model (the uncertainty set $\mathcal{U}$ is the box $B$), where we ignore the stochastic nature of the perturbations and their independence. This model yields the Interval Robust Counterpart (9.16);

- The ellipsoidal uncertainty model ($\mathcal{U}$ is the ellipsoid $E(\Omega)$), which takes into account the stochastic nature of data perturbations and yields the Ellipsoidal Robust Counterpart (9.17).

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

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

$$
\begin{aligned}
l &\leq \sum_{j=1}^{n} a_j^{\mathrm{n}} x_j - \sum_{j=1}^{n} \sigma_j y_j - \Omega \sqrt{\sum_{j=1}^{n} \sigma_j^2 u_j^2}, \\
&\quad \sum_{j=1}^{n} a_j^{\mathrm{n}} x_j + \sum_{j=1}^{n} \sigma_j z_j + \Omega \sqrt{\sum_{j=1}^{n} \sigma_j^2 v_j^2} \leq u, \\
-y_j &\leq x_j - u_j \leq y_j, j = 1, ..., n, \\
-z_j &\leq x_j - v_j \leq z_j, j = 1, ..., n.
\end{aligned}
\tag{9.21}
$$

Second, it turns out that our intuition is correct: As a model of uncertainty, $U(\Omega)$ is as reliable as the ellipsoid $E(\Omega)$. Specifically, if $x$ can be extended to a feasible solution of (9.21), then the probability for $x$ to satisfy a realization of (9.15) is $\geq 1 - \exp\{-\Omega^2/2\}$.

The conclusion is that if we have reasons to assume that the perturbations of uncertain coefficients in a constraint of an uncertain linear optimization problem are (a) random, (b) independent of each other, and (c) symmetrically distributed in given intervals, then it makes sense to associate with this constraint an interval-ellipsoidal model of uncertainty and use a system of linear and conic quadratic inequalities (9.21). Please note that when building the robust counterpart of an uncertain linear optimization problem, one can use different models of the uncertainty (e.g., interval, ellipsoidal, combined interval-ellipsoidal) for different uncertain constraints within the same problem.

### 9.1.5 Further references

For further information about robust linear optimization consult [6], [7].

## 9.2 Geometric (posynomial) optimization

### 9.2.1 The problem

A *geometric optimization* problem can be stated as follows

$$\begin{aligned}
\text{minimize} \quad & \sum_{k \in J_0} c_k \prod_{j=0}^{n-1} t_j^{a_{kj}} \\
\text{subject to} \quad & \sum_{k \in J_i} c_k \prod_{j=0}^{n-1} t_j^{a_{kj}} \quad \leq \quad 1, \quad i = 1, \ldots, m, \\
& t > 0,
\end{aligned} \tag{9.22}$$

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*. In general, the problem (9.22) is very hard to solve. However, the posynomial case where it is required that

$$c > 0$$

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

$$t_j = e^{x_j} \tag{9.23}$$

we obtain the problem

$$\begin{aligned}
\text{minimize} \quad & \sum_{k \in J_0} c_k e^{\sum_{j=0}^{n-1} a_{kj} x_j} \\
\text{subject to} \quad & \sum_{k \in J_i} c_k e^{\sum_{j=0}^{n-1} a_{kj} x_j} \quad \leq \quad 1, \quad i = 1, \ldots, m,
\end{aligned} \tag{9.24}$$

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

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

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

As stated, the problem (9.24) is non-separable. However, using

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

we obtain the separable problem

$$
\begin{array}{lrcll}
\text{minimize} & \displaystyle\sum_{t \in J_0} e^{v_t} & & & \\[2ex]
\text{subject to} & \displaystyle\sum_{t \in J_i} e^{v_t} & \leq & 1, & i = 1, \ldots, m, \\[2ex]
& \displaystyle\sum_{j=0}^{n-1} a_{tj} x_j - v_t & = & -\mathtt{log}(c_t), & t = 0, \ldots, T,
\end{array}
\tag{9.25}
$$

which is a separable convex optimization problem.

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

## 9.2.2   Applications

A large number of practical applications, particularly in electrical circuit design, can be cast as a geometric optimization problem. We will not review these applications here but rather refer the reader to [8] and the references therein.

## 9.2.3   Modeling tricks

A lot of tricks that can be used for modeling posynomial optimization problems are described in [8]. Therefore, in this section we cover only one important case.

### 9.2.3.1   Equalities

In general, equalities are not allowed in (9.22), i.e.

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

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

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

of a monomial equality. The equality is identical to

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

which in turn is identical to the two inequalities

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

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

## 9.2.4   Problematic formulations

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

- The solution vector is finite, but an optimal objective value can only be a approximated.

- The optimal objective value is finite but implies that a variable in the solution is infinite.

### 9.2.4.1   Finite unattainable solution

The following problem illustrates an unattainable solution:

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

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

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

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

Now, the issue should be clear: If a variable $x$ appears only with nonnegative exponents, then fixing $x = 0$ will minimize all terms in which it appears — but such a solution cannot be attained.

### 9.2.4.2   Infinite solution

A similar problem will occur if a finite optimal objective value requires a variable to be infinite. This can be illustrated by the following example:

$$\begin{array}{rccc} \text{minimize} & x^{-2} \\ \text{subject to} & x^{-1} & \leq & 1, \\ & x > 0, \end{array}$$

which is a valid geometric programming problem. In this case the optimal objective is 0, but this requires $x = \infty$, which is unattainable.

Again, this specific case will appear if a variable $x$ appears only with negative exponents in the problem, implying that each term in which it appears can be minimized for $x \to \infty$.

### 9.2.5   An example

Consider the example

$$\begin{array}{rccc} \text{minimize} & x^{-1}y \\ \text{subject to} & x^2 y^{-\frac{1}{2}} + 3y^{\frac{1}{2}}z^{-1} & \leq & 1, \\ & xy^{-1} & = & z^2, \\ & -x & \leq & -\dfrac{1}{10}, \\ & x & \leq & 3, \\ & x, y, z > 0, \end{array}$$

which is not a geometric optimization problem. However, using the obvious transformations we obtain the problem

$$\begin{array}{rccc} \text{minimize} & x^{-1}y \\ \text{subject to} & x^2 y^{-\frac{1}{2}} + 3y^{\frac{1}{2}}z^{-1} & \leq & 1, \\ & xy^{-1}z^{-2} & \leq & 1, \\ & x^{-1}yz^2 & \leq & 1, \\ & \dfrac{1}{10}x^{-1} & \leq & 1, \\ & \dfrac{1}{3}x & \leq & 1, \\ & x, y, z > 0, \end{array} \qquad (9.26)$$

which is a geometric optimization problem.

### 9.2.6   Solving the example

The problem (9.26) can be defined and solved in the MOSEK toolbox as shown below.

────────────────────────────[ go2.m ]────────────────────────────

```
1   % go2.m
2
3   c     = [1 1 3 1 1 0.1 1/3]';
4   a     = sparse([[-1  1  0];
5                   [2 -0.5 0];
6                   [0 0.5 -1];
```

```
7                      [1 -1 -2];
8                      [-1 1 2];
9                      [-1 0 0];
10                     [1 0 0]]);
11
12  map   = [0 1 1 2 3 4 5]';
13  [res] = mskgpopt(c,a,map);
14
15  fprintf('\nPrimal optimal solution to original gp:');
16  fprintf(' %e',exp(res.sol.itr.xx));
17  fprintf('\n\n');
18
19  % Compute the optimal objective value and
20  % the constraint activities.
21  v = c.*exp(a*res.sol.itr.xx);
22
23  % Add appropriate terms together.
24  f = sparse(map+1,1:7,ones(size(map)))*v;
25
26  % First objective value. Then constraint values.
27  fprintf('Objective value: %e\n',log(f(1)));
28  fprintf('Constraint values:');
29  fprintf(' %e',log(f(2:end)));
30  fprintf('\n\n');
31
32  % Dual multipliers (should be negative)
33  fprintf('Dual variables (should be negative):');
34  fprintf(' %e',res.sol.itr.y);
35  fprintf('\n\n');
```

### 9.2.7 Exporting to a file

It's possible to write a geometric optimization problem to a file with the command:

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

This file format is compatible with the `mskexpopt` command line tool. See the MOSEK Tools User's manual for details on `mskexpopt`. This file format can be useful for sending debug information to MOSEK or for testing. It's also possible to read the above format with the command:

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

### 9.2.8 Further information

More information about geometric optimization problems is located in [3], [9], [8].

# Chapter 10

# 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.

## 10.1   How an optimizer works

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

Presolve:

   Preprocessing to reduce the size of the problem.

Dualizer:

   Choosing whether to solve the primal or the dual form of the problem.

Scaling:

   Scaling the problem for better numerical stability.

Optimize:

   Solve the problem using selected method.

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

## 10.1.1    Presolve

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

- remove redundant constraints,

- eliminate fixed variables,

- remove linear dependencies,

- substitute out (implied) free variables, and

- reduce the size of the optimization problem in general.

After the presolved problem has been optimized the solution is automatically postsolved so that the returned solution is valid for the original problem. Hence, the presolve is completely transparent. For further details about the presolve phase, please see [10], [11].

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.

### 10.1.1.1    Numerical issues in the presolve

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

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

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

### 10.1.1.2 Eliminator

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

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

$y$ is an implied free variable that can be substituted out of the problem, if deemed worthwhile.

If the eliminator consumes too much time or memory compared to the reduction in problem size gained it may be disabled. This can be done with the parameter MSK_IPAR_PRESOLVE_ELIMINATOR_USE to MSK_OFF.

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

### 10.1.1.3 Linear dependency checker

The purpose of the linear dependency check is to remove linear dependencies among the linear equalities. For instance, the three linear equalities

$$
\begin{aligned}
x_1 + x_2 + x_3 &= 1, \\
x_1 + 0.5x_2 &= 0.5, \\
0.5x_2 + x_3 &= 0.5
\end{aligned}
$$

contain exactly one linear dependency. This implies that one of the constraints can be dropped without changing the set of feasible solutions. Removing linear dependencies is in general a good idea since it reduces the size of the problem. Moreover, the linear dependencies are likely to introduce numerical problems in the optimization phase.

It is best practise to build models without linear dependencies. If the linear dependencies are removed at the modeling stage, the linear dependency check can safely be disabled by setting the parameter MSK_IPAR_PRESOLVE_LINDEP_USE to MSK_OFF.

## 10.1.2 Dualizer

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

- MSK_IPAR_INTPNT_SOLVE_FORM: In case of the interior-point optimizer.

- MSK_IPAR_SIM_SOLVE_FORM: In case of the simplex optimizer.

Note that currently only linear problems may be dualized.

### 10.1.3   Scaling

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

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

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

### 10.1.4   Using multiple threads

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

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

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

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

## 10.2    Linear optimization

### 10.2.1    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`.

## 10.2.2 The interior-point optimizer

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

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

$$
\begin{array}{lll}
\text{minimize} & c^T x \\
\text{subject to} & Ax & = & b, \\
& x \geq 0.
\end{array}
\tag{10.1}
$$

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

Since it is not known beforehand whether problem (10.1) has an optimal solution, is primal infeasible or is dual infeasible, the optimization algorithm must deal with all three situations. This is the reason that MOSEK solves the so-called homogeneous model

$$
\begin{array}{rcl}
Ax - b\tau & = & 0, \\
A^T y + s - c\tau & = & 0, \\
-c^T x + b^T y - \kappa & = & 0, \\
x, s, \tau, \kappa & \geq & 0,
\end{array}
\tag{10.2}
$$

where $y$ and $s$ correspond to the dual variables in (10.1), and $\tau$ and $\kappa$ are two additional scalar variables. Note that the homogeneous model (10.2) always has solution since

$$
(x, y, s, \tau, \kappa) = (0, 0, 0, 0, 0)
$$

is a solution, although not a very interesting one.

Any solution

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

to the homogeneous model (10.2) satisfies

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

Moreover, there is always a solution that has the property

$$
\tau^* + \kappa^* > 0.
$$

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

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

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

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

is a primal-dual optimal solution.

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

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

This implies that at least one of

$$
- c^T x^* > 0 \tag{10.3}
$$

or

$$
b^T y^* > 0 \tag{10.4}
$$

is satisfied. If (10.3) is satisfied then $x^*$ is a certificate of dual infeasibility, whereas if (10.4) is satisfied then $y^*$ is a certificate of dual infeasibility.

In summary, by computing an appropriate solution to the homogeneous model, all information required for a solution to the original problem is obtained. A solution to the homogeneous model can be computed using a primal-dual interior-point algorithm [12].

### 10.2.2.1   Interior-point termination criterion

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

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

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

to homogeneous model is generated where

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

Whenever the trial solution satisfies the criterion

$$
\begin{aligned}
\left\| A\frac{x^k}{\tau^k} - b \right\|_\infty &\leq \epsilon_p(1 + \|b\|_\infty), \\
\left\| A^T\frac{y^k}{\tau^k} + \frac{s^k}{\tau^k} - c \right\|_\infty &\leq \epsilon_d(1 + \|c\|_\infty), \text{ and} \\
\min\left( \frac{(x^k)^T s^k}{(\tau^k)^2}, \left| \frac{c^T x^k}{\tau^k} - \frac{b^T y^k}{\tau^k} \right| \right) &\leq \epsilon_g\max\left( 1, \frac{\min(|c^T x^k|, |b^T y^k|)}{\tau^k} \right),
\end{aligned}
\tag{10.5}
$$

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 (10.5) 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

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

It is easy to verify that

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

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

| Tolerance | Parameter name |
|-----------|----------------|
| $\epsilon_p$ | MSK_DPAR_INTPNT_TOL_PFEAS |
| $\epsilon_d$ | MSK_DPAR_INTPNT_TOL_DFEAS |
| $\epsilon_g$ | MSK_DPAR_INTPNT_TOL_REL_GAP |
| $\epsilon_i$ | MSK_DPAR_INTPNT_TOL_INFEAS |

Table 10.1: Parameters employed in termination criterion.

Finally, if

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

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

It is possible to adjust the tolerances $\epsilon_p$, $\epsilon_d$, $\epsilon_g$ and $\epsilon_i$ using parameters; see table 10.1 for details. The default values of the termination tolerances are chosen such that for a majority of problems appearing in practice it is not possible to achieve much better accuracy. Therefore, tightening the tolerances usually is not worthwhile. However, an inspection of (10.5) 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 [12]. 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, $\epsilon_p$, $\epsilon_d$ and $\epsilon_g$, has to be relaxed together to achieve an effect.

In some cases the interior-point method terminates having found a solution not too far from meeting the optimality condition (10.5). A solution is defined as *near optimal* if scaling $\epsilon_p$, $\epsilon_d$ and $\epsilon_g$ by any number $\epsilon_n \in [1.0, +\infty]$ conditions (10.5) 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 10.2.2.2 requires an optimal solution to work well; hence basis identification should turned off if the termination criterion is relaxed.

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

### 10.2.2.2   Basis identification

An interior-point optimizer does not return an optimal basic solution unless the problem has a unique primal and dual optimal solution. Therefore, the interior-point optimizer has an optional post-processing step that computes an optimal basic solution starting from the optimal interior-point solution. More information about the basis identification procedure may be found in [13].

Please note that a basic solution is often more accurate than an interior-point solution.

By default MOSEK performs a basis identification. However, if a basic solution is not needed, the

basis identification procedure can be turned off. The parameters

- MSK_IPAR_INTPNT_BASIS,

- MSK_IPAR_BI_IGNORE_MAX_ITER, and

- MSK_IPAR_BI_IGNORE_NUM_ERROR

controls when basis identification is performed.

### 10.2.2.3 The interior-point log

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

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

The first line displays the number of threads used by the optimizer and second line tells that the optimizer choose to solve the dual problem rather than the primal problem. The next line displays the problem dimensions as seen by the optimizer, and the "Factor..." lines show various statistics. This is followed by the iteration log.

Using the same notation as in section 10.2.2 the columns of the iteration log has the following meaning:

- ITE: Iteration index.

- PFEAS: $\left\lVert Ax^k - b\tau^k \right\rVert_\infty$. The numbers in this column should converge monotonically towards to zero but may stall at low level due to rounding errors.

- DFEAS: $\left\lVert A^T y^k + s^k - c\tau^k \right\rVert_\infty$. The numbers in this column should converge monotonically toward to zero but may stall at low level due to rounding errors.

- GFEAS: $\left\lVert -cx^k + b^T y^k - \kappa^k \right\rVert_\infty$. The numbers in this column should converge monotonically toward to zero but may stall at low level due to rounding errors.

- PRSTATUS: This number converge to 1 if the problem has an optimal solution whereas it converge to $-1$ if that is not the case.

- **POBJ**: $c^T x^k / \tau^k$. An estimate for the primal objective value.

- **DOBJ**: $b^T y^k / \tau^k$. An estimate for the dual objective value.

- **MU**: $\frac{(x^k)^T s^k + \tau^k \kappa^k}{n+1}$ . The numbers in this column should always converge monotonically to zero.

- **TIME**: Time spend since the optimization started.

### 10.2.3 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 10.2.4 for a discussion.

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

#### 10.2.3.1 Simplex termination criterion

The simplex optimizer terminates when it finds an optimal basic solution or an infeasibility certificate. A basic solution is optimal when it is primal and dual feasible; see (17.1) and (17.2) for a definition of the primal and dual problem. Due the fact that to computations are performed in finite precision MOSEK allows violation of primal and dual feasibility within certain tolerances. The user can control the allowed primal and dual infeasibility with the parameters MSK_DPAR_BASIS_TOL_X and MSK_DPAR_BASIS_TOL_S.

#### 10.2.3.2 Starting from an existing solution

When using the simplex optimizer it may be possible to reuse an existing solution and thereby reduce the solution time significantly. When a simplex optimizer starts from an existing solution it is said to perform a *hot-start*. If the user is solving a sequence of optimization problems by solving the problem, making modifications, and solving again, MOSEK will hot-start automatically.

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

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

#### 10.2.3.3 Numerical difficulties in the simplex optimizers

Though MOSEK is designed to minimize numerical instability, completely avoiding it is impossible when working in finite precision. MOSEK counts a "numerical unexpected behavior" event inside the optimizer as a *set-back*. The user can define how many set-backs the optimizer accepts; if that number

is exceeded, the optimization will be aborted. Set-backs are implemented to avoid long sequences where the optimizer tries to recover from an unstable situation.

Set-backs are, for example, repeated singularities when factorizing the basis matrix, repeated loss of feasibility, degeneracy problems (no progress in objective) and other events indicating numerical difficulties. If the simplex optimizer encounters a lot of set-backs the problem is usually badly scaled; in such a situation try to reformulate into a better scaled problem. Then, if a lot of set-backs still occur, trying one or more of the following suggestions may be worthwhile:

- Raise tolerances for allowed primal or dual feasibility: Hence, increase the value of

  - MSK_DPAR_BASIS_TOL_X, and
  - MSK_DPAR_BASIS_TOL_S.

- Raise or lower pivot tolerance: Change the MSK_DPAR_SIMPLEX_ABS_TOL_PIV parameter.

- Switch optimizer: Try another optimizer.

- Switch off crash: Set both MSK_IPAR_SIM_PRIMAL_CRASH and MSK_IPAR_SIM_DUAL_CRASH to 0.

- Experiment with other pricing strategies: Try different values for the parameters

  - MSK_IPAR_SIM_PRIMAL_SELECTION and
  - MSK_IPAR_SIM_DUAL_SELECTION.

- If you are using hot-starts, in rare cases switching off this feature may improve stability. This is controlled by the MSK_IPAR_SIM_HOTSTART parameter.

- Increase maximum set-backs allowed controlled by MSK_IPAR_SIM_MAX_NUM_SETBACKS.

- If the problem repeatedly becomes infeasible try switching off the special degeneracy handling. See the parameter MSK_IPAR_SIM_DEGEN for details.

## 10.2.4 The interior-point or the simplex optimizer?

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

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

## 10.2.5 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.

## 10.3   Linear network optimization

### 10.3.1   Network flow problems

Linear optimization problems with network flow structure can often be solved significantly faster with a specialized version of the simplex method [14] than with the general solvers.

MOSEK includes a network simplex solver which frequently solves network problems significantly faster than the standard simplex optimizers.

To use the network simplex optimizer, do the following:

- Input the network flow problem as an ordinary linear optimization problem.

- Set the parameters

    - `MSK_IPAR_OPTIMIZER` to `MSK_OPTIMIZER_NETWORK_PRIMAL_SIMPLEX`.

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

## 10.4   Conic optimization

### 10.4.1   The interior-point optimizer

For conic optimization problems only an interior-point type optimizer is available. The interior-point optimizer is an implementation of the so-called homogeneous and self-dual algorithm. For a detailed description of the algorithm, please see [15].

#### 10.4.1.1   Interior-point termination criteria

The parameters controlling when the conic interior-point optimizer terminates are shown in Table 10.2.

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

Table 10.2: Parameters employed in termination criterion.

## 10.5 Nonlinear convex optimization

### 10.5.1 The interior-point optimizer

For quadratic, quadratically constrained, and general convex optimization problems an interior-point type optimizer is available. The interior-point optimizer is an implementation of the homogeneous and self-dual algorithm. For a detailed description of the algorithm, please see [16], [17].

#### 10.5.1.1 The convexity requirement

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

The following parameters are available to control the convexity check:

- MSK_IPAR_CHECK_CONVEXITY: Turn convexity check on/off.

- MSK_DPAR_CHECK_CONVEXITY_REL_TOL: Tolerance for convexity check.

- MSK_IPAR_LOG_CHECK_CONVEXITY: Turn on more log information for debugging.

#### 10.5.1.2 The differentiabilty requirement

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

For instance, the function

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

is differentiable everywhere whereas the function

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

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

| Parameter name | Purpose |
|---|---|
| MSK_DPAR_INTPNT_NL_TOL_PFEAS | Controls primal feasibility |
| MSK_DPAR_INTPNT_NL_TOL_DFEAS | Controls dual feasibility |
| MSK_DPAR_INTPNT_NL_TOL_REL_GAP | Controls relative gap |
| MSK_DPAR_INTPNT_TOL_INFEAS | Controls when the problem is declared infeasible |
| MSK_DPAR_INTPNT_NL_TOL_MU_RED | Controls when the complementarity is reduced enough |

Table 10.3: Parameters employed in termination criteria.

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

$$x \geq 0$$

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

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

$$f(x) = \frac{1}{x} + \frac{1}{1-x}. \tag{10.6}$$

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, (10.6) can be reformulated as follows

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

### 10.5.1.3    Interior-point termination criteria

The parameters controlling when the general convex interior-point optimizer terminates are shown in Table 10.3.

# 10.6 Solving problems in parallel

If a computer has multiple CPUs, or has a CPU with multiple cores, it is possible for MOSEK to take advantage of this to speed up solution times.

## 10.6.1 Thread safety

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

## 10.6.2 The parallelized interior-point optimizer

The interior-point optimizer is capable of using multiple CPUs or cores. This implies that whenever the MOSEK interior-point optimizer solves an optimization problem, it will try to divide the work so that each core gets a share of the work. The user decides how many coress MOSEK should exploit.

It is not always possible to divide the work equally, and often parts of the computations and the coordination of the work is processed sequentially, even if several cores are present. Therefore, the speed-up obtained when using multiple cores is highly problem dependent. However, as a rule of thumb, if the problem solves very quickly, i.e. in less than 60 seconds, it is not advantageous to use the parallel option.

The `MSK_IPAR_NUM_THREADS` parameter sets the number of threads (and therefore the number of cores) that the interior point optimizer will use.

## 10.6.3 The concurrent optimizer

An alternative to the parallel interior-point optimizer is the *concurrent optimizer*. The idea of the concurrent optimizer is to run multiple optimizers on the same problem concurrently, for instance, it allows you to apply the interior-point and the dual simplex optimizers to a linear optimization problem concurrently. The concurrent optimizer terminates when the first of the applied optimizers has terminated successfully, and it reports the solution of the fastest optimizer. In that way a new optimizer has been created which essentially performs as the fastest of the interior-point and the dual simplex optimizers. Hence, the concurrent optimizer is the best one to use if there are multiple optimizers available in MOSEK for the problem and you cannot say beforehand which one will be faster.

Note in particular that any solution present in the task will also be used for hot-starting the simplex algorithms. One possible scenario would therefore be running a hot-start dual simplex in parallel with interior point, taking advantage of both the stability of the interior-point method and the ability of the simplex method to use an initial solution.

By setting the

    `MSK_IPAR_OPTIMIZER`

parameter to

| Optimizer | Associated parameter | Default priority |
|---|---|---|
| MSK_OPTIMIZER_INTPNT | MSK_IPAR_CONCURRENT_PRIORITY_INTPNT | 4 |
| MSK_OPTIMIZER_FREE_SIMPLEX | MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX | 3 |
| MSK_OPTIMIZER_PRIMAL_SIMPLEX | MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX | 2 |
| MSK_OPTIMIZER_DUAL_SIMPLEX | MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX | 1 |

Table 10.4: Default priorities for optimizer selection in concurrent optimization.

MSK_OPTIMIZER_CONCURRENT

the concurrent optimizer chosen.

The number of optimizers used in parallel is determined by the

MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS.

parameter. Moreover, the optimizers are selected according to a preassigned priority with optimizers having the highest priority being selected first. The default priority for each optimizer is shown in Table 10.6.3. For example, setting the MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS parameter to 2 tells the concurrent optimizer to the apply the two optimizers with highest priorities: In the default case that means the interior-point optimizer and one of the simplex optimizers.

### 10.6.3.1   Concurrent optimization through the API

The following example shows how to call the concurrent optimizer through the API.

# Chapter 11

# The solution summary

All computations inside MOSEK are performed using finite precision floating point numbers. This implies the reported solution isonly be an approximate optimal solution. Therefore after solving an optimization problem it is important to investigate how good an approximation the solution is. Recall for a convex optimization problem 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.

Thus the solution summary reports information that makes it possible to evaluate the quality of the solution obtained.

Suppose we have solved a problem as

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

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

```
Basic solution summary
  Problem status  : PRIMAL_AND_DUAL_FEASIBLE
  Solution status : OPTIMAL
  Primal.  obj: -4.6475314286e+002  Viol.  con: 2e-014   var: 0e+000
  Dual.    obj: -4.6475316001e+002  Viol.  con: 7e-009   var: 4e-016
```

The summary reports information for the basic solution. In this case we see:

- The problem status is primal and dual feasible which means the problem has an optimal solution. The problem status can be obtained using `res.sol.itr.prosta`.

- The solution status is optimal. The solution status can be obtained using `res.sol.itr.solsta`.

- Next information about the primal solution is reported. The information consists of the objective value and violation meassures for the primal solution. In this case violations for the constraints and variables are small meaning the solution is very close to being an exact feasible solution. The primal feasibility measure is reported in `res.info.`<span style="color:red">MSK_DINF_INTPNT_PRIMAL_FEAS</span>.

- Similarly for the dual solution the violations are small and hence the dual solution is feasible. The dual feasibility measure is reported in `res.info.MSK_DINF_INTPNT_DUAL_FEAS`.

- Finally, it can be seen that the primal and dual objective values are almost identical. Using `res.info.MSK_DINF_INTPNT_PRIMAL_OBJ` and `res.info.MSK_DINF_INTPNT_DUAL_OBJ` the primal and dual objective values can be obtained.

To summarize in this case a primal and a dual solution with small feasiblity violations are available. Moreover, the primal and dual objective values are almost identical and hence it can be concluded that the reported solution is a good approximation to the optimal solution.

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

```
Basic solution summary
  Problem status  : PRIMAL_INFEASIBLE
  Solution status : PRIMAL_INFEASIBLE_CER
  Dual.    obj: 3.5894503823e+004   Viol.  con: 0e+000   var: 2e-008
```

i.e. MOSEK reports that the solution is a certificate of primal infeasibility. Since the problem is primal infeasible it does not make sense to report any information about the primal solution. However, the dual solution should be a certificate of the primal infeasibility. If the problem is a minimization problem then the dual objective value should be positive and in the case of a maximization problem it should be negative. The quality of the certificate can be evaluated by comparing the dual objective value to the violations. Indeed if the objective value is large compared to the largest violation then the certificate highly accurate. Here is an example

```
Basic solution summary
  Problem status  : PRIMAL_INFEASIBLE
  Solution status : PRIMAL_INFEASIBLE_CER
  Dual.    obj: 3.0056574100e-005   Viol.  con: 9e-013   var: 2e-011
```

of a not so strong infeasibility certificate because the dual objective value is small compared to largest violation.

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: -1.4500853392e+001  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 negative and the objective should be large compared to the worst violation if the certificate is strong.

# Chapter 12

# The optimizers for mixed-integer problems

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

- linear,

- quadratic and quadratically constrained, and

- conic

problems.

Readers unfamiliar with integer optimization are recommended to consult some relevant literature, e.g. the book [5] by Wolsey.

## 12.1 Some concepts and facts related to mixed-integer optimization

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

In practice this implies that the focus should be on computing a near optimal solution quickly rather than at locating an optimal solution. Even if the problem is only solved approximately, it is important to know how far the approximate solution is from an optimal one. In order to say something about the goodness of an approximate solution then the concept of a relaxation is important.

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

Table 12.1: Mixed-integer optimizers.

The mixed-integer optimization problem

$$
\begin{aligned}
z^* \quad = \quad &\text{minimize} \quad && c^T x \\
&\text{subject to} \quad && Ax \quad = \quad b, \\
& && x \geq 0 \\
& && x_j \in \mathbb{Z}, \qquad \forall j \in \mathcal{J},
\end{aligned}
\tag{12.1}
$$

has the continuous relaxation

$$
\begin{aligned}
\underline{z} \quad = \quad &\text{minimize} \quad && c^T x \\
&\text{subject to} \quad && Ax \quad = \quad b, \\
& && x \geq 0
\end{aligned}
\tag{12.2}
$$

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

There are two important observations about the continuous relaxation. Firstly, the continuous relaxation is usually much faster to optimize than the mixed-integer problem. Secondly if $\hat{x}$ is any feasible solution to (12.1) and

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

then

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

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

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

## 12.2   The mixed-integer optimizers

MOSEK includes two mixed-integer optimizers which are compared in Table 12.1. Both optimizers can handle problems with linear, quadratic objective and constraints and conic constraints. However, a problem must not contain both quadratic objective and constraints and conic constraints.

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

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

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

## 12.3 The mixed-integer conic optimizer

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

The mixed-integer conic employs three phases:

Presolve:

> In this phase the optimizer tries to reduce the size of the problem using preprocessing techniques. Moreover, it strengthens the continuous relaxation, if possible.

Heuristic:

> Using heuristics the optimizer tries to guess a good feasible solution.

Optimization:

> The optimal solution is located using a variant of the branch-and-cut method.

### 12.3.1 Presolve

In the preprocessing stage redundant variables and constraints are removed. The presolve stage can be turned off using the `MSK_IPAR_MIO_PRESOLVE_USE` parameter.

### 12.3.2 Heuristic

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

### 12.3.3    The optimization phase

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

### 12.3.4    Caveats

The mixed-integer conic optimizer ignores the parameter

MSK_IPAR_MIO_CONT_SOL:

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

## 12.4    The mixed-integer optimizer

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

The process of solving an integer optimization problem can be split in three phases:

Presolve:

>   In this phase the optimizer tries to reduce the size of the problem using preprocessing techniques. Moreover, it strengthens the continuous relaxation, if possible.

Heuristic:

>   Using heuristics the optimizer tries to guess a good feasible solution.

Optimization:

>   The optimal solution is located using a variant of the branch-and-cut method.

### 12.4.1    Presolve

In the preprocessing stage redundant variables and constraints are removed. The presolve stage can be turned off using the MSK_IPAR_MIO_PRESOLVE_USE parameter.

### 12.4.2    Heuristic

Initially, the integer optimizer tries to guess a good feasible solution using different heuristics:

- First a very simple rounding heuristic is employed.

- Next, if deemed worthwhile, the *feasibility pump* heuristic is used.

- Finally, if the two previous stages did not produce a good initial solution, more sophisticated heuristics are used.

The following parameters can be used to control the effort made by the integer optimizer to find an initial feasible solution.

- **MSK_IPAR_MIO_HEURISTIC_LEVEL**: Controls how sophisticated and computationally expensive a heuristic to employ.

- **MSK_DPAR_MIO_HEURISTIC_TIME**: The minimum amount of time to spend in the heuristic search.

- **MSK_IPAR_MIO_FEASPUMP_LEVEL**: Controls how aggressively the feasibility pump heuristic is used.

### 12.4.3 The optimization phase

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

## 12.5 Termination criterion

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

A candidate solution that is feasible to the continuous relaxation is said to be an integer feasible solution if the criterion

$$\min(|x_j| - \lfloor x_j \rfloor, \lceil x_j \rceil - |x_j|) \leq \max(\delta_1, \delta_2 |x_j|) \ \forall j \in \mathcal{J}$$

is satisfied.

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

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

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

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

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

### 12.5.1 Relaxed termination

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

| Tolerance | Parameter name |
|-----------|----------------|
| $\delta_1$ | MSK_DPAR_MIO_TOL_ABS_RELAX_INT |
| $\delta_2$ | MSK_DPAR_MIO_TOL_REL_RELAX_INT |
| $\delta_3$ | MSK_DPAR_MIO_TOL_ABS_GAP |
| $\delta_4$ | MSK_DPAR_MIO_TOL_REL_GAP |
| $\delta_5$ | MSK_DPAR_MIO_NEAR_TOL_ABS_GAP |
| $\delta_6$ | MSK_DPAR_MIO_NEAR_TOL_REL_GAP |

Table 12.2: Integer optimizer tolerances.

| Parameter name | Delayed | Explanation |
|----------------|---------|-------------|
| MSK_IPAR_MIO_MAX_NUM_BRANCHES | Yes | Maximum number of branches allowed. |
| MSK_IPAR_MIO_MAX_NUM_RELAXS | Yes | Maximum number of relaxations allowed. |
| MSK_IPAR_MIO_MAX_NUM_SOLUTIONS | Yes | Maximum number of feasible integer solutions allowed. |

Table 12.3: Parameters affecting the termination of the integer optimizer.

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

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

### 12.5.2   Important parameters

All $\delta$ tolerances can be adjusted using suitable parameters — see Table 12.2. In Table 12.3 some other parameters affecting the integer optimizer termination criterion are shown. Please note that if the effect of a parameter is delayed, the associated termination criterion is applied only after some time, specified by the MSK_DPAR_MIO_DISABLE_TERM_TIME parameter.

## 12.6   How to speed up the solution process

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

- Relax the termination criterion: In case the run time is not acceptable, the first thing to do is to relax the termination criterion — see Section 12.5 for details.

- Specify a good initial solution: In many cases a good feasible solution is either known or easily computed using problem specific knowledge. If a good feasible solution is known, it is usually worthwhile to use this as a starting point for the integer optimizer.

- Improve the formulation: A mixed-integer optimization problem may be impossible to solve in one form and quite easy in another form. However, it is beyond the scope of this manual to discuss good formulations for mixed-integer problems. For discussions on this topic see for example [5].

## 12.7 Understanding solution quality

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

- The solution status key returned by MOSEK.

- The *optimality gap*: A measure for how much the located solution can deviate from the optimal solution to the problem.

- Feasibility. How much the solution violates the constraints of the problem.

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

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

The objective value of the solution is guarantied to be within $\epsilon$ 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 13

# The solution summary for mixed integer problems

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

```
Integer solution solution summary
  Problem status  : PRIMAL_FEASIBLE
  Solution status : INTEGER_OPTIMAL
  Primal.  obj: 4.0593518000e+005   Viol.  con: 4e-015   var: 3e-014   itg: 3e-014
```

The main diffrence compared to continous 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 higly 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.

# Chapter 14

# The analyzers

## 14.1 The problem analyzer

The problem analyzer prints a detailed survey of the

- linear constraints and objective

- quadratic constraints

- conic constraints

- variables

of the model.

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

The problem analyzer is run using the `mosekopt('anapro')` command and produces something similar to the following (this is the problemanalyzer's survey of the `aflow30a` problem from the MIPLIB 2003 collection, see Appendix 26 for more examples):

```
Analyzing the problem

Constraints                Bounds                    Variables
 upper bd:       421        ranged  : all             cont:       421
 fixed   :       58                                   bin :       421


 ----------------------------------------------------------------------------

Objective, min cx
   range: min |c|: 0.00000   min |c|>0: 11.0000     max |c|: 500.000
 distrib:        |c|          vars
                  0            421
```

157

```
            [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.


### 14.1.1   General characteristics

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

Constraints

upper bd:
The number of upper bounded constraints, $\sum_{j=0}^{n-1} a_{ij}x_j \leq u_i^c$

lower bd:
The number of lower bounded constraints, $l_i^c \leq \sum_{j=0}^{n-1} a_{ij}x_j$

ranged :
The number of ranged constraints, $l_i^c \leq \sum_{j=0}^{n-1} a_{ij}x_j \leq u_i^c$

fixed :
The number of fixed constraints, $l_i^c = \sum_{j=0}^{n-1} a_{ij}x_j = u_i^c$

free :
The number of free constraints

Bounds

upper bd:
The number of upper bounded variables, $x_j \leq u_j^x$

lower bd:
The number of lower bounded variables, $l_k^x \leq x_j$

ranged :
The number of ranged variables, $l_k^x \leq x_j \leq u_j^x$

fixed :
The number of fixed variables, $l_k^x = x_j = u_j^x$

free :
The number of free variables

Variables

cont:
The number of continuous variables, $x_j \in \mathbb{R}$

bin :
The number of binary variables, $x_j \in \{0, 1\}$

int :
The number of general integer variables, $x_j \in \mathbb{Z}$

Only constraints, bounds and domains actually in the model will be reported on, cf. appendix 26; if all entities in a section turn out to be of the same kind, the number will be replaced by `all` for brevity.

## 14.1.2   Objective

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

The range is displayed using three terms:

`min |c|:`
>    The minimum absolute value among all coeffecients

`min |c|>0:`
>    The minimum absolute value among the nonzero coefficients

`max |c|:`
>    The maximum absolute value among the coefficients

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

- If `min |c|` is greater than zero, the `min |c|?0` term is obsolete and will not be displayed

- If only one or two different coefficients occur this will be displayed using `all` and an explicit listing of the coefficients

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

## 14.1.3   Linear constraints

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

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

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

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

corresponding constraints have any quadratic terms or the corresponding variables are used in conic or quadratic constraints; cf. the last two examples of appendix 26.

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

### 14.1.4 Constraint and variable bounds

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

### 14.1.5 Quadratic constraints

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

Note: Quadratic constraints may also have a linear part, but that will be included in the linear constraints survey; this means that if a problem has one or more pure quadratic constraints, part three of the survey will report an equal number of linear constraint rows with 0 (zero) nonzeros, cf. the last example in appendix 26. Likewise, variables that appear in quadratic terms only will be reported as empty columns (0 nonzeros) in the linear constraint report.

### 14.1.6 Conic constraints

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

## 14.2 Analyzing infeasible problems

When developing and implementing a new optimization model, the first attempts will often be either infeasible, due to specification of inconsistent constraints, or unbounded, if important constraints have been left out.

In this chapter we will

- go over an example demonstrating how to locate infeasible constraints using the MOSEK infeasibility report tool,

- discuss in more general terms which properties that may cause infeasibilities, and

- present the more formal theory of infeasible and unbounded problems.
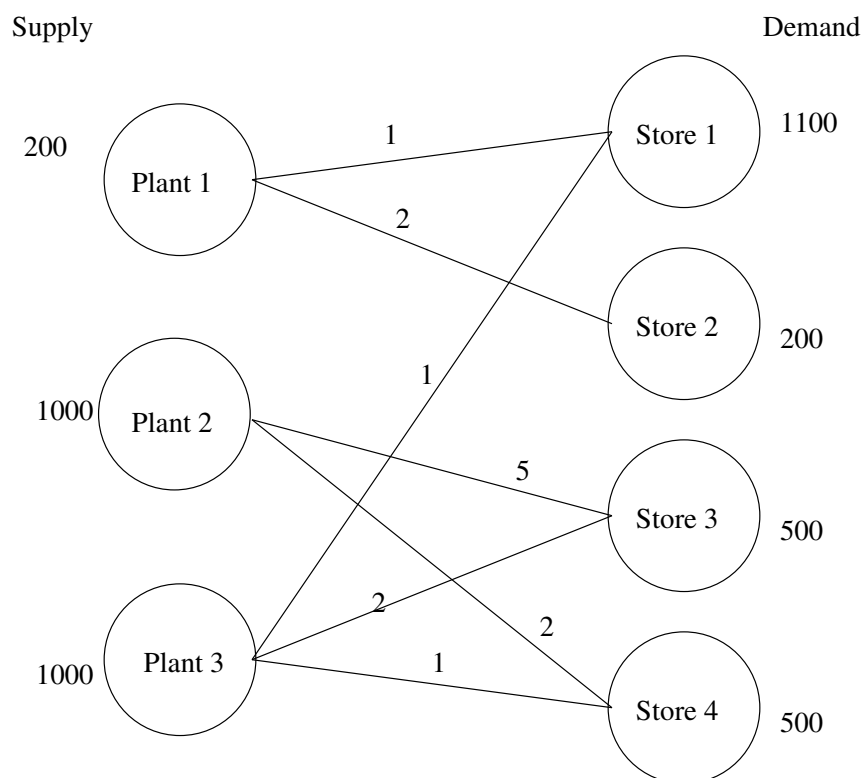
Supply                                                   Demand



Figure 14.1: Supply, demand and cost of transportation.

## 14.2.1   Example: Primal infeasibility

A problem is said to be *primal infeasible* if no solution exists that satisfy all the constraints of the problem.

As an example of a primal infeasible problem consider the problem of minimizing the cost of transportation between a number of production plants and stores: Each plant produces a fixed number of goods, and each store has a fixed demand that must be met. Supply, demand and cost of transportation per unit are given in figure 14.1. The problem represented in figure 14.1 is infeasible, since the total demand

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

exceeds the total supply

$$2200 = 200 + 1000 + 1000$$

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

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

$$(14.1)$$

Solving the problem (14.1) using MOSEK will result in a solution, a solution status and a problem status. Among the log output from the execution of MOSEK on the above problem are the lines:

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

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

## 14.2.2 Locating the cause of primal infeasibility

Usually a primal infeasible problem status is caused by a mistake in formulating the problem and therefore the question arises: "What is the cause of the infeasible status?" When trying to answer this question, it is often advantageous to follow these steps:

- Remove the objective function. This does not change the infeasible status but simplifies the problem, eliminating any possibility of problems related to the objective function.

- Consider whether your problem has some necessary conditions for feasibility and examine if these are satisfied, e.g. total supply should be greater than or equal to total demand.

- Verify that coefficients and bounds are reasonably sized in your problem.

If the problem is still primal infeasible, some of the constraints must be relaxed or removed completely. The MOSEK infeasibility report (Section 14.2.4) may assist you in finding the constraints causing the infeasibility.

Possible ways of relaxing your problem include:

- Increasing (decreasing) upper (lower) bounds on variables and constraints.

- Removing suspected constraints from the problem.

Returning to the transportation example, we discover that removing the fifth constraint

$$x_{12} = 200$$

makes the problem feasible.

### 14.2.3   Locating the cause of dual infeasibility

A problem may also be *dual infeasible*. In this case the primal problem is often unbounded, mening that feasbile solutions exists such that the objective tends towards infinity. An example of a dual infeasible and primal unbounded problem is:

$$
\begin{aligned}
\text{minimize} \quad & x_1 \\
\text{subject to} \quad & x_1 \leq 5.
\end{aligned}
$$

To resolve a dual infeasibility the primal problem must be made more restricted by

- Adding upper or lower bounds on variables or constraints.

- Removing variables.

- Changing the objective.

#### 14.2.3.1   A cautious note

The problem

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

is clearly infeasible. Moreover, if any one of the constraints are dropped, then the problem becomes feasible.

This illustrates the worst case scenario that all, or at least a significant portion, of the constraints are involved in the infeasibility. Hence, it may not always be easy or possible to pinpoint a few constraints which are causing the infeasibility.

### 14.2.4   The infeasibility report

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

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

### 14.2.4.1 Example: Primal infeasibility

We will reuse the example (14.1) located in `infeas.lp`:

```
\
\ An example of an infeasible linear problem.
\
minimize
 obj: + 1 x11 + 2 x12 + 1 x13
      + 4 x21 + 2 x22 + 5 x23
      + 4 x31 + 1 x32 + 2 x33
 st
  s0: + x11 + x12        <= 200
  s1: + x23 + x24        <= 1000
  s2: + x31 +x33 + x34 <= 1000
  d1: + x11 + x31        = 1100
  d2: + x12              = 200
  d3: + x23 + x33        = 500
  d4: + x24 + x34        = 500
 bounds
 end
```

Using the command line (please remeber it accepts options following the C API format)

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

MOSEK produces the following infeasibility report

```
MOSEK PRIMAL INFEASIBILITY REPORT.

Problem status: The problem is primal infeasible

The following constraints are involved in the primal infeasibility.

Index   Name    Lower bound     Upper bound     Dual lower      Dual upper
0       s0      NONE            2.000000e+002   0.000000e+000   1.000000e+000
2       s2      NONE            1.000000e+003   0.000000e+000   1.000000e+000
3       d1      1.100000e+003   1.100000e+003   1.000000e+000   0.000000e+000
4       d2      2.000000e+002   2.000000e+002   1.000000e+000   0.000000e+000

The following bound constraints are involved in the infeasibility.

Index   Name    Lower bound     Upper bound     Dual lower      Dual upper
8       x33     0.000000e+000   NONE            1.000000e+000   0.000000e+000
10      x34     0.000000e+000   NONE            1.000000e+000   0.000000e+000
```

The infeasibility report is divided into two sections where the first section shows which constraints that are important for the infeasibility. In this case the important constraints are the ones named `s0`, `s2`, `d1`, and `d2`. The values in the columns "`Dual lower`" and "`Dual upper`" are also useful, since a non-zero *dual lower* value for a constraint implies that the lower bound on the constraint is important for the infeasibility. Similarly, a non-zero *dual upper* value implies that the upper bound on the constraint is important for the infeasibility.

It is also possible to obtain the infeasible subproblem. The command line

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

produces the files `rinfeas.bas.inf.lp`. In this case the content of the file `rinfeas.bas.inf.lp` is

```
minimize
```

```
 Obj: + CFIXVAR
st
 s0: + x11 + x12 <= 200
 s2: + x31 + x33 + x34 <= 1e+003
 d1: + x11 + x31 = 1.1e+003
 d2: + x12 = 200
bounds
 x11 free
 x12 free
 x13 free
 x21 free
 x22 free
 x23 free
 x31 free
 x32 free
 x24 free
 CFIXVAR = 0e+000
end
```

which is an optimization problem. This problem is identical to (14.1), except that the objective and some of the constraints and bounds have been removed. Executing the command

```
mosek -d MSK_IPAR_INFEAS_REPORT_AUTO MSK_ON infeas.bas.inf.lp
```

demonstrates that the reduced problem is **primal infeasible**. Since the reduced problem is usually smaller than original problem, it should be easier to locate the cause of the infeasibility in this rather than in the original (14.1).

### 14.2.4.2   Example: Dual infeasibility

The example problem

```
maximize -  200 y1 - 1000 y2 - 1000 y3
         - 1100 y4 -  200 y5 -  500 y6
         -  500 y7
subject to
   x11: y1+y4 < 1
   x12: y1+y5 < 2
   x23: y2+y6 < 5
   x24: y2+y7 < 2
   x31: y3+y4 < 1
   x33: y3+y6 < 2
   x44: y3+y7 < 1
bounds
   y1 < 0
   y2 < 0
   y3 < 0
   y4 free
   y5 free
   y6 free
   y7 free
end
```

is dual infeasible. This can be verified by proving that

```
 y1=-1, y2=-1, y3=0, y4=1, y5=1
```

is a certificate of dual infeasibility.  In this example the following infeasibility report is produced

(slightly edited):

```
The following constraints are involved in the infeasibility.

Index    Name            Activity          Objective        Lower bound      Upper bound
0        x11             -1.000000e+00                       NONE             1.000000e+00
4        x31             -1.000000e+00                       NONE             1.000000e+00

The following variables are involved in the infeasibility.

Index    Name            Activity          Objective        Lower bound      Upper bound
3        y4              -1.000000e+00     -1.100000e+03     NONE             NONE
Interior-point solution
Problem status  : DUAL_INFEASIBLE
Solution status : DUAL_INFEASIBLE_CER
Primal - objective: 1.1000000000e+03   eq. infeas.: 0.00e+00 max bound infeas.: 0.00e+00 cone infeas.: 0.00e+00
Dual   - objective: 0.0000000000e+00   eq. infeas.: 0.00e+00 max bound infeas.: 0.00e+00 cone infeas.: 0.00e+00
```

Let $x^*$ denote the reported primal solution. MOSEK states

- that the problem is *dual infeasible*,

- that the reported solution is a certificate of dual infeasibility, and

- that the infeasibility measure for $x^*$ is approximately zero.

Since it was an maximization problem, this implies that

$$c^t x^* > 0. \tag{14.2}$$

For a minimization problem this inequality would have been reversed — see (14.5).

From the infeasibility report we see that the variable y4, and the constraints x11 and x33 are involved in the infeasibility since these appear with non-zero values in the "Activity" column.

One possible strategy to "fix" the infeasibility is to modify the problem so that the certificate of infeasibility becomes invalid. In this case we may do one the following things:

- Put a lower bound in y3. This will directly invalidate the certificate of dual infeasibility.

- Increase the object coefficient of y3. Changing the coefficients sufficiently will invalidate the inequality (14.2) and thus the certificate.

- Put lower bounds on x11 or x31. This will directly invalidate the certificate of infeasibility.

Please note that modifying the problem to invalidate the reported certificate does *not* imply that the problem becomes dual feasible — the infeasibility may simply "move", resulting in a new infeasibility.

More often, the reported certificate can be used to give a hint about errors or inconsistencies in the model that produced the problem.

## 14.2.5    Theory concerning infeasible problems

This section discusses the theory of infeasibility certificates and how MOSEK uses a certificate to produce an infeasibility report. In general, MOSEK solves the problem

$$
\begin{array}{llcccc}
\text{minimize} & & c^T x + c^f & & \\
\text{subject to} & l^c & \leq & Ax & \leq & u^c, \\
& l^x & \leq & x & \leq & u^x
\end{array}
\tag{14.3}
$$

where the corresponding dual problem is

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

We use the convension that for any bound that is not finite, the corresponding dual variable is fixed at zero (and thus will have no influence on the dual problem). For example

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

## 14.2.6    The certificate of primal infeasibility

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

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

with a positive objective value. That is, $(s_l^{c*}, s_u^{c*}, s_l^{x*}, s_u^{x*})$ is a certificate of primal infeasibility if

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

and

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

The well-known Farkas Lemma tells us that (14.3) is infeasible if and only if a certificate of primal infeasibility exists.

Let $(s_l^{c*}, s_u^{c*}, s_l^{x*}, s_u^{x*})$ be a certificate of primal infeasibility then

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

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

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

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

### 14.2.7 The certificate of dual infeasibility

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

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

with negative objective value, where we use the definitions

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

and

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

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

$$
\begin{array}{rcccl}
& & c^T x^* & < & 0, \\
\bar{l}^c & \leq & Ax^* & \leq & \bar{u}^c, \\
\bar{l}^x & \leq & x^* & \leq & \bar{u}^x
\end{array}
\tag{14.5}
$$

The well-known Farkas Lemma tells us that (14.4) is infeasible if and only if a certificate of dual infeasibility exists.

Note that if $x^*$ is a certificate of dual infeasibility then for any $j$ such that

$$x_j^* \neq 0,$$

variable $j$ is involved in the dual infeasibility.

# Chapter 15

# Primal feasibility repair

Section 14.2.2 discusses how MOSEK treats infeasible problems. In particular, it is discussed which information MOSEK returns when a problem is infeasible and how this information can be used to pinpoint the cause of the infeasibility.

In this section we discuss how to repair a primal infeasible problem by relaxing the constraints in a controlled way. For the sake of simplicity we discuss the method in the context of linear optimization.

## 15.1 Manual repair

Subsequently we discuss an automatic method for repairing an infeasible optimization problem. However, it should be observed that the best way to repair an infeasible problem usually depends on what the optimization problem models. For instance in many optimization problem it does not make sense to relax the constraints $x \geq 0$ e.g. it is not possible to produce a negative quantity. Hence, whatever automatic method MOSEK provides it will never be as good as a method that exploits knowledge about what is being modelled. This implies that it is usually better to remove the underlying cause of infeasibility at the modelling stage.

Indeed consider the example

$$
\begin{array}{llrcrcrcrcl}
\text{minimize} & & & & & & & & & & \\
\text{subject to} & & x_1 & + & x_2 & & & & & = & 1, \\
& & & & & & x_3 & + & x_4 & = & 1, \\
& - & x_1 & & & - & x_3 & & & = & -1+\epsilon \\
& & & - & x_2 & & & - & x_4 & = & -1, \\
& & x_1, & & x_2, & & x_3, & & x_4 & \geq & 0
\end{array}
\tag{15.1}
$$

then if we add the equalties together we obtain the implied equality

$$
0 = \epsilon
$$

which is infeasible for any $\epsilon \neq 0$. Here the infeasibility is caused by a linear dependency in the constraint matrix and that the right-hand side does not match if $\epsilon \neq 0$. Observe even if the problem is feasible then just a tiny perturbation to the right-hand side will make the problem infeasible. Therefore, even though the problem can be repaired then a much more robust solution is to avoid problems with linear dependent constraints. Indeed if a problem contains linear dependencies then the problem is either infeasible or contains redundant constraints. In the above case any of the equality constraints can be removed while not changing the set of feasible solutions.

To summarize linear dependencies in the constraints can give rise to infeasible problems and therefore it is better to avoid them. Note that most network flow models usually is formulated with one linear dependent constraint.

Next consider the problem

$$
\begin{array}{lrcl}
\text{minimize} & & & \\
\text{subject to} & x_1 - 0.01x_2 & = & 0 \\
& x_2 - 0.01x_3 & = & 0 \\
& x_3 - 0.01x_4 & = & 0 \\
& x_1 & \geq & -1.0e-9 \\
& x_1 & \leq & 1.0e-9 \\
& x_4 & \leq & -1.0e-4
\end{array}
\tag{15.2}
$$

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

$$-1.0e-9 \leq x_1 \leq 1.0e-9$$

are tight then the MOSEK presolve will fix variable $x_1$ at the mid point between the bounds i.e. at 0. It easy to see that this implies $x_4 = 0$ too which leads to the incorrect conclusion that the problem is infeasible. Observe tiny change of the size 1.0e-9 make the problem switch from feasible to infeasible. Such a problem is inherently unstable and is hard to solve. We normally call such a problem ill-posed. In general it is recommended to avoid ill-posed problems, but if that is not possible then one solution to this issue is is to reduce the parameter to say `MSK_DPAR_PRESOLVE_TOL_X` to say 1.0e-10. This will at least make sure that the presolve does not make the wrong conclusion.

## 15.2   Automatic repair

In this section we will describe the idea behind a method that automatically can repair an infeasible probem. The main idea can be described as follows.

Consider the linear optimization problem with $m$ constraints and $n$ variables

$$
\begin{array}{lrcccl}
\text{minimize} & & & c^T x + c^f & & \\
\text{subject to} & l^c & \leq & Ax & \leq & u^c, \\
& l^x & \leq & x & \leq & u^x,
\end{array}
\tag{15.3}
$$

which is assumed to be infeasible.

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

$$
\begin{array}{rllcl}
\text{minimize} & & p(v_l^c, v_u^c, v_l^x, v_u^x) & & \\
\text{subject to} & l^c \leq & Ax + v_l^c - v_u^c & \leq & u^c, \\
& l^x \leq & x + v_l^x - v_u^x & \leq & u^x, \\
& & v_l^c, v_u^c, v_l^x, v_u^x \geq 0 & &
\end{array}
\tag{15.4}
$$

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

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

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

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

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

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

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

then a natural choice is

$$
p(v_l^c, v_u^c, v_l^x, v_u^x) = (w_l^c)^T v_l^c + (w_u^c)^T v_u^c + (w_l^x)^T v_l^x + (w_u^x)^T v_u^x.
\tag{15.5}
$$

Hence, the penalty function $p()$ is a weighted sum of the relaxation and therefore the problem (15.4) keeps the amount of relaxation at a minimum. Please observe that

- the problem (15.6) is always feasible.

- a negative weight implies problem (15.6) is unbounded. For this reason if the value of a weight is negative MOSEK fixes the associated elasticity variable to zero. Clearly, if one or more of the weights are negative may imply that it is not possible repair the problem.

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

## 15.2.1 Caveats

Observe if the infeasible problem

$$\begin{array}{llrcr}
\text{minimize} & x + z \\
\text{subject to} & x & = & -1, \\
& x & \geq & 0
\end{array} \qquad (15.6)$$

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

Another and more important caveat is that only a minimial repair is perfomed i.e. the repair that just make the problem feasible. Hence, the repaired problem is barely feasible and that sometimes make the repaired problem hard to solve.


## 15.3  Feasibility repair in MOSEK

MOSEK includes a function that repair an infeasible problem using the idea described in the previous section simply by passing a set of weights to MOSEK. This can be used for linear and conic optimization problems, possibly having integer constrained variables.


### 15.3.1  An example using feasibility repair

Consider the example linear optimization

$$\begin{array}{lrcrcr}
\text{minimize} & -10x_1 & & -9x_2, \\
\text{subject to} & 7/10x_1 & + & 1x_2 & \leq & 630, \\
& 1/2x_1 & + & 5/6x_2 & \leq & 600, \\
& 1x_1 & + & 2/3x_2 & \leq & 708, \\
& 1/10x_1 & + & 1/4x_2 & \leq & 135, \\
& x_1, & & x_2 & \geq & 0, \\
& & x_2 \geq 650
\end{array} \qquad (15.7)$$

which is infeasible. Now suppose we wish to use MOSEK to suggest a modification to the bounds that makes the problem feasible.

The following example

```
[r,res]=mosekopt('read(feasrepair.lp)');
res.prob.primalrepair = [];
res.prob.primalrepair.wux = [1,1];
res.prob.primalrepair.wlx = [1,1];
res.prob.primalrepair.wuc = [1,1,1,1];
res.prob.primalrepair.wlc = [1,1,1,1];

param.MSK_IPAR_LOG_FEAS_REPAIR = 3;
[r,res]=mosekopt('minimize primalrepair',res.prob,param);
```

will form the repaired problem and solve it. The parameter

```
MSK_IPAR_LOG_FEAS_REPAIR
```

controls the amount of log output from the repair. A value of 2 causes the optimal repair to printed out. If the fields `wlx`, `wux`, `wlc` or `wuc` are not specified, they are all assumed to be 1-vectors of appropriate dimensions.

The output from running the commands above is:

```
Copyright (c) 1998-2013 MOSEK ApS, Denmark. WWW: http://mosek.com

Open file 'feasrepair.lp'

Read summary
  Type            : LO (linear optimization problem)
  Objective sense : min
  Constraints     : 4
  Scalar variables : 2
  Matrix variables : 0
  Time            : 0.0

Computer
  Platform              : Windows/64-X86
  Cores                 : 4

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

Primal feasibility repair started.
Optimizer started.
Interior-point optimizer started.
Presolve started.
Linear dependency checker started.
Linear dependency checker terminated.
Eliminator started.
Total number of eliminations : 2
Eliminator terminated.
Eliminator - tries               : 1              time               : 0.00
Eliminator - elim's              : 2
Lin. dep.  - tries               : 1              time               : 0.00
Lin. dep.  - number              : 0
Presolve terminated. Time: 0.00
Optimizer  - threads             : 1
Optimizer  - solved problem      : the primal
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              : 5.40e+001
ITE PFEAS    DFEAS    GFEAS    PRSTATUS   POBJ               DOBJ                MU       TIME
0   2.7e+001 1.0e+000 4.8e+000 1.00e+000  4.195228609e+000   0.000000000e+000    1.0e+000 0.00
1   2.4e+001 8.6e-001 1.5e+000 0.00e+000  1.227497414e+001   1.504971820e+001    2.6e+000 0.00
2   2.6e+000 9.7e-002 1.7e-001 -6.19e-001 4.363064729e+001   4.648523094e+001    3.0e-001 0.00
3   4.7e-001 1.7e-002 3.1e-002 1.24e+000  4.256803136e+001   4.298540657e+001    5.2e-002 0.00
4   8.7e-004 3.2e-005 5.7e-005 1.08e+000  4.249989892e+001   4.250078747e+001    9.7e-005 0.00
```

```
5   8.7e-008 3.2e-009 5.7e-009 1.00e+000  4.249999999e+001  4.250000008e+001  9.7e-009 0.00
6   8.7e-012 3.2e-013 5.7e-013 1.00e+000  4.250000000e+001  4.250000000e+001  9.7e-013 0.00
Basis identification started.
Primal basis identification phase started.
ITER      TIME
0         0.00
Primal basis identification phase terminated. Time: 0.00
Dual basis identification phase started.
ITER      TIME
0         0.00
Dual basis identification phase terminated. Time: 0.00
Basis identification terminated. Time: 0.00
Interior-point optimizer terminated. Time: 0.00.


Optimizer terminated. Time: 0.03
Basic solution summary
  Problem status  : PRIMAL_AND_DUAL_FEASIBLE
  Solution status : OPTIMAL
  Primal.  obj: 4.2500000000e+001   Viol.  con: 1e-013   var: 0e+000
  Dual.    obj: 4.2500000000e+001   Viol.  con: 0e+000   var: 5e-013
Optimal objective value of the penalty problem: 4.250000000000e+001


Repairing bounds.
Increasing the upper bound -2.25e+001 on constraint 'c4' (3) with 1.35e+002.
Decreasing the lower bound 6.50e+002 on variable 'x2' (4) with 2.00e+001.
Primal feasibility repair terminated.
Optimizer started.
Interior-point optimizer started.
Presolve started.
Presolve terminated. Time: 0.00
Interior-point optimizer terminated. Time: 0.00.


Optimizer terminated. Time: 0.00


Interior-point solution summary
  Problem status  : PRIMAL_AND_DUAL_FEASIBLE
  Solution status : OPTIMAL
  Primal.  obj: -5.6700000000e+003 Viol.  con: 0e+000   var: 0e+000
  Dual.    obj: -5.6700000000e+003 Viol.  con: 0e+000   var: 0e+000

Basic solution summary
  Problem status  : PRIMAL_AND_DUAL_FEASIBLE
  Solution status : OPTIMAL
  Primal.  obj: -5.6700000000e+003  Viol.  con: 0e+000   var: 0e+000
  Dual.    obj: -5.6700000000e+003  Viol.  con: 0e+000   var: 0e+000

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

```
    Primal-dual simplex   - iterations : 0         time: 0.00
   Mixed integer           - relaxations: 0        time: 0.00
```

reports the optimal repair. In this case it is to increase the upper bound on constraint `c4` by 1.35e2 and decrease the lower bound on variable `x2` by 20.

# Chapter 16

# Sensitivity analysis

## 16.1  Introduction

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

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

## 16.2  Restrictions

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

## 16.3  References

The book [1] discusses the classical sensitivity analysis in Chapter 10 whereas the book [18] presents a modern introduction to sensitivity analysis. Finally, it is recommended to read the short paper [19] to avoid some of the pitfalls associated with sensitivity analysis.
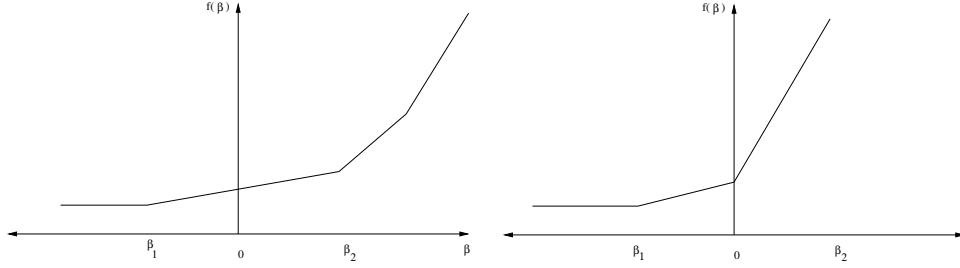
Figure 16.1:   The optimal value function $f_{l_i^c}(\beta)$. Left: $\beta = 0$ is in the interior of linearity interval. Right: $\beta = 0$ is a breakpoint.

## 16.4    Sensitivity analysis for linear problems

### 16.4.1    The optimal objective value function

Assume that we are given the problem

$$
\begin{aligned}
z(l^c, u^c, l^x, u^x, c) \quad = \quad & \text{minimize} && c^T x \\
& \text{subject to} \quad l^c \quad \leq \quad && Ax && \leq \quad u^c, \\
& && l^x \leq x \leq u^x,
\end{aligned}
\tag{16.1}
$$

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

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

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

$$
f_{l_i^c}(\beta)
\tag{16.2}
$$

shows the optimal objective value as a function of $\beta$. Please note that a change in $\beta$ corresponds to a perturbation in $l_i^c$ and hence (16.2) shows the optimal objective value as a function of $l_i^c$.

It is possible to prove that the function (16.2) is a piecewise linear and convex function, i.e. the function may look like the illustration in Figure 16.1. Clearly, if the function $f_{l_i^c}(\beta)$ does not change much when $\beta$ is changed, then we can conclude that the optimal objective value is insensitive to changes in $l_i^c$. Therefore, we are interested in the rate of change in $f_{l_i^c}(\beta)$ for small changes in $\beta$ — specificly the gradient

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

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

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

for which

$$f'_{l^c_i}(\beta) = f'_{l^c_i}(0).$$

Since $f_{l^c_i}$ is not a smooth function $f'_{l^c_i}$ may not be defined at 0, as illustrated by the right example in figure 16.1. In this case we can define a left and a right shadow price and a left and a right linearity interval.

The function $f_{l^c_i}$ considered only changes in $l^c_i$. We can define similar functions for the remaining parameters of the $z$ defined in (16.1) as well:

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

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

### 16.4.1.1  Equality constraints

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

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

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

## 16.4.2  The basis type sensitivity analysis

The classical sensitivity analysis discussed in most textbooks about linear optimization, e.g. [1], is based on an optimal basic solution or, equivalently, on an optimal basis. This method may produce misleading results [18] but is **computationally cheap**. Therefore, and for historical reasons this method is available in MOSEK We will now briefly discuss the basis type sensitivity analysis. Given an optimal basic solution which provides a partition of variables into basic and non-basic variables, the basis type sensitivity analysis computes the linearity interval $[\beta_1, \beta_2]$ so that the basis remains optimal for the perturbed problem. A shadow price associated with the linearity interval is also computed. However, it is well-known that an optimal basic solution may not be unique and therefore the result depends on the optimal basic solution employed in the sensitivity analysis. This implies that the computed interval is only a subset of the largest interval for which the shadow price is constant. Furthermore, the optimal objective value function might have a breakpoint for $\beta = 0$. In this case the basis type sensitivity method will only provide a subset of either the left or the right linearity interval.

In summary, the basis type sensitivity analysis is computationally cheap but does not provide complete information. Hence, the results of the basis type sensitivity analysis should be used with care.

### 16.4.3   The optimal partition type sensitivity analysis

Another method for computing the complete linearity interval is called the *optimal partition type sensitivity analysis*. The main drawback of the optimal partition type sensitivity analysis is that it is computationally expensive compared to the basis type analysts. This type of sensitivity analysis is currently provided as an experimental feature in MOSEK.

Given the optimal primal and dual solutions to (16.1), i.e. $x^*$ and $((s_l^c)^*, (s_u^c)^*, (s_l^x)^*, (s_u^x)^*)$ the optimal objective value is given by

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

The left and right shadow prices $\sigma_1$ and $\sigma_2$ for $l_i^c$ are given by this pair of optimization problems:

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

and

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

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

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

and

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

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

The left and right shadow prices for $c_j$ denoted $\sigma_1$ and $\sigma_2$ respectively are computed as follows:

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

and

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

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

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

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

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

and

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

## 16.4.4 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 Figure 16.2. 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 minimize

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

subject to
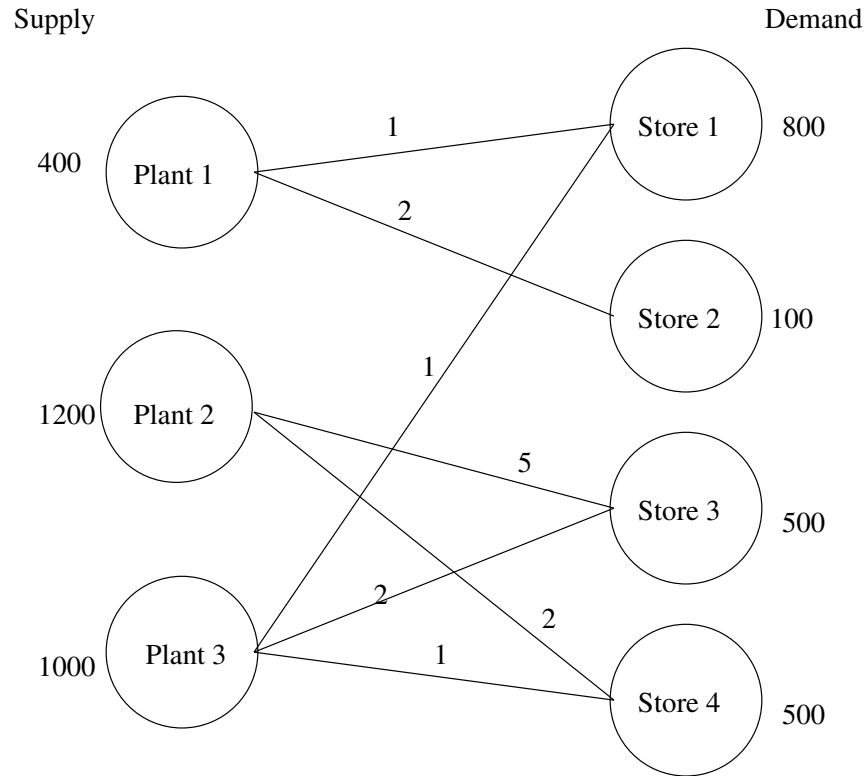
Figure 16.2: Supply, demand and cost of transportation.

**Basis type**

| Con. | $\beta_1$ | $\beta_2$ | $\sigma_1$ | $\sigma_2$ |
|---|---|---|---|---|
| 1 | −300.00 | 0.00 | 3.00 | 3.00 |
| 2 | −700.00 | +∞ | 0.00 | 0.00 |
| 3 | −500.00 | 0.00 | 3.00 | 3.00 |
| 4 | −0.00 | 500.00 | 4.00 | 4.00 |
| 5 | −0.00 | 300.00 | 5.00 | 5.00 |
| 6 | −0.00 | 700.00 | 5.00 | 5.00 |
| 7 | −500.00 | 700.00 | 2.00 | 2.00 |
| Var. | $\beta_1$ | $\beta_2$ | $\sigma_1$ | $\sigma_2$ |
| $x_{11}$ | −∞ | 300.00 | 0.00 | 0.00 |
| $x_{12}$ | −∞ | 100.00 | 0.00 | 0.00 |
| $x_{23}$ | −∞ | 0.00 | 0.00 | 0.00 |
| $x_{24}$ | −∞ | 500.00 | 0.00 | 0.00 |
| $x_{31}$ | −∞ | 500.00 | 0.00 | 0.00 |
| $x_{33}$ | −∞ | 500.00 | 0.00 | 0.00 |
| $x_{34}$ | −0.000000 | 500.00 | 2.00 | 2.00 |

**Optimal partition type**

| Con. | $\beta_1$ | $\beta_2$ | $\sigma_1$ | $\sigma_2$ |
|---|---|---|---|---|
| 1 | −300.00 | 500.00 | 3.00 | 1.00 |
| 2 | −700.00 | +∞ | −0.00 | −0.00 |
| 3 | −500.00 | 500.00 | 3.00 | 1.00 |
| 4 | −500.00 | 500.00 | 2.00 | 4.00 |
| 5 | −100.00 | 300.00 | 3.00 | 5.00 |
| 6 | −500.00 | 700.00 | 3.00 | 5.00 |
| 7 | −500.00 | 700.00 | 2.00 | 2.00 |
| Var. | $\beta_1$ | $\beta_2$ | $\sigma_1$ | $\sigma_2$ |
| $x_{11}$ | −∞ | 300.00 | 0.00 | 0.00 |
| $x_{12}$ | −∞ | 100.00 | 0.00 | 0.00 |
| $x_{23}$ | −∞ | 500.00 | 0.00 | 2.00 |
| $x_{24}$ | −∞ | 500.00 | 0.00 | 0.00 |
| $x_{31}$ | −∞ | 500.00 | 0.00 | 0.00 |
| $x_{33}$ | −∞ | 500.00 | 0.00 | 0.00 |
| $x_{34}$ | −∞ | 500.00 | 0.00 | 2.00 |

Table 16.1: Ranges and shadow prices related to bounds on constraints and variables. Left: Results for the basis type sensitivity analysis. Right: Results for the optimal partition type sensitivity analysis.

$$
\begin{array}{rcccccccl}
x_{11} & + & x_{12} & & & & & & \leq & 400, \\
& & & x_{23} & + & x_{24} & & & \leq & 1200, \\
& & & & & & x_{31} + x_{33} + x_{34} & & \leq & 1000, \\
x_{11} & & & & & + & x_{31} & & = & 800, \\
& & x_{12} & & & & & & = & 100, \\
& & & x_{23} & + & & x_{33} & & = & 500, \\
& & & & & x_{24} + & x_{34} & & = & 500, \\
x_{11}, & & x_{12}, & x_{23}, & & x_{24}, & x_{31}, \quad x_{33}, \quad x_{34} & & \geq & 0.
\end{array}
\tag{16.3}
$$

The basis type and the optimal partition type sensitivity results for the transportation problem are shown in Table 16.1 and 16.2 respectively. Examining the results from the optimal partition type sensitivity analysis we see that for constraint number 1 we have $\sigma_1 \neq \sigma_2$ and $\beta_1 \neq \beta_2$. Therefore, we have a left linearity interval of $[-300, 0]$ and a right interval of $[0, 500]$. The corresponding left and right shadow 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

**Basis type**

| Var. | $\beta_1$ | $\beta_2$ | $\sigma_1$ | $\sigma_2$ |
|------|-----------|-----------|------------|------------|
| $c_1$ | $-\infty$ | 3.00 | 300.00 | 300.00 |
| $c_2$ | $-\infty$ | $\infty$ | 100.00 | 100.00 |
| $c_3$ | $-2.00$ | $\infty$ | 0.00 | 0.00 |
| $c_4$ | $-\infty$ | 2.00 | 500.00 | 500.00 |
| $c_5$ | $-3.00$ | $\infty$ | 500.00 | 500.00 |
| $c_6$ | $-\infty$ | 2.00 | 500.00 | 500.00 |
| $c_7$ | $-2.00$ | $\infty$ | 0.00 | 0.00 |

**Optimal partition type**

| Var. | $\beta_1$ | $\beta_2$ | $\sigma_1$ | $\sigma_2$ |
|------|-----------|-----------|------------|------------|
| $c_1$ | $-\infty$ | 3.00 | 300.00 | 300.00 |
| $c_2$ | $-\infty$ | $\infty$ | 100.00 | 100.00 |
| $c_3$ | $-2.00$ | $\infty$ | 0.00 | 0.00 |
| $c_4$ | $-\infty$ | 2.00 | 500.00 | 500.00 |
| $c_5$ | $-3.00$ | $\infty$ | 500.00 | 500.00 |
| $c_6$ | $-\infty$ | 2.00 | 500.00 | 500.00 |
| $c_7$ | $-2.00$ | $\infty$ | 0.00 | 0.00 |

Table 16.2:   Ranges and shadow prices related to the objective coefficients. Left: Results for the basis type sensitivity analysis. Right: Results for the optimal partition type sensitivity analysis.

$$\beta \in [0, 300]$$

then the optimal objective value will increase by the value

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

## 16.5   Sensitivity analysis in the MATLAB toolbox

The following describe sensitivity analysis from the MATLAB toolbox.

### 16.5.1   On bounds

The index of bounds/variables to analyzed for sensitivity are specified in the following subfields of the matlab structure `prob`:

`.prisen.cons.subu`

>   Indexes of constraints, where upper bounds are analyzed for sensitivity.

`.prisen.cons.subl`

>   Indexes of constraints, where lower bounds are analyzed for sensitivity.

`.prisen.vars.subu`

>   Indexes of variables, where upper bounds are analyzed for sensitivity.

`.prisen.vars.subl`

>   Indexes of variables, where lower bounds are analyzed for sensitivity.

`.duasen.sub`

>   Index of variables where coefficients are 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`.

### 16.5.1.1   prisen

The field `prisen` is structured as follows:

`.cons`

  MATLAB structure with subfields:

  `.lr_bl`

    Left value $\beta_1$ in the linearity interval for a lower bound.

  `.rr_bl`

    Right value $\beta_2$ in the linearity interval for a lower bound.

  `.ls_bl`

    Left shadow price $s_l$ for a lower bound.

  `.rs_bl`

    Right shadow price $s_r$ for a lower bound.

  `.lr_bu`

    Left value $\beta_1$ in the linearity interval for an upper bound.

  `.rr_bu`

    Right value $\beta_2$ in the linearity interval for an upper bound.

  `.ls_bu`

    Left shadow price $s_l$ for an upper bound.

  `.rs_bu`

    Right shadow price $s_r$ for an upper bound.

`.var`

  MATLAB structure with subfields:

  `.lr_bl`

    Left value $\beta_1$ in the linearity interval for a lower bound on a varable.

  `.rr_bl`

    Right value $\beta_2$ in the linearity interval for a lower bound on a varable.

  `.ls_bl`

    Left shadow price $s_l$ for a lower bound on a varable.

  `.rs_bl`

    Right shadow price $s_r$ for lower bound on a varable.

.lr_bu

> Left value $\beta_1$ in the linearity interval for an upper bound on a varable.

.rr_bu

> Right value $\beta_2$ in the linearity interval for an upper bound on a varable.

.ls_bu

> Left shadow price $s_l$ for an upper bound on a varables.

.rs_bu

> Right shadow price $s_r$ for an upper bound on a varables.

### 16.5.1.2   duasen

The field `duasen` is structured as follows:

.lr_c

> Left value $\beta_1$ of linearity interval for an objective coefficient.

.rr_c

> Right value $\beta_2$ of linearity interval for an objective coefficient.

.ls_c

> Left shadow price $s_l$ for an objective coefficients .

.rs_c

> Right shadow price $s_r$ for an objective coefficients.

## 16.5.2   Selecting analysis type

The type (basis or optimal partition) of analysis to be performed can be selected by setting the parameter

```
MSK_IPAR_SENSITIVITY_TYPE
```

to one of the values:

```
MSK_SENSITIVITY_TYPE_BASIS = 0
MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION = 1
```

as seen in the following example.

## 16.5.3   An example

Consider the problem defined in (16.3). 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.

```
% sensitivity.m
```

```matlab
% Obtain all symbolic constants
% defined by MOSEK.
[r,res]  = mosekopt('symbcon');
sc       = res.symbcon;
[r,res] = mosekopt('read(../../tools/examples/data/transport.lp)');
prob = res.prob;
% analyse upper bound 1:7
prob.prisen.cons.subl = [];
prob.prisen.cons.subu = [1:7];
% analyse lower bound on variables 1:7
prob.prisen.vars.subl = [1:7];
prob.prisen.vars.subu = [];
% analyse 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
```

```matlab
  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 `sensitivity.m` is shown below.

```
Basis sensitivity results:

Sensitivity for bounds on constraints:
con = 1, beta_1 = -300.0, beta_2 = 0.0, delta_1 = 3.0,delta_2 = 3.0
con = 2, beta_1 = -700.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0
con = 3, beta_1 = -500.0, beta_2 = 0.0, delta_1 = 3.0,delta_2 = 3.0
con = 4, beta_1 = -0.0, beta_2 = 500.0, delta_1 = 4.0,delta_2 = 4.0
con = 5, beta_1 = -0.0, beta_2 = 300.0, delta_1 = 5.0,delta_2 = 5.0
con = 6, beta_1 = -0.0, beta_2 = 700.0, delta_1 = 5.0,delta_2 = 5.0
con = 7, beta_1 = -500.0, beta_2 = 700.0, delta_1 = 2.0,delta_2 = 2.0
Sensitivity for bounds on variables:
var = 1, beta_1 = Inf, beta_2 = 300.0, delta_1 = 0.0,delta_2 = 0.0
var = 2, beta_1 = Inf, beta_2 = 100.0, delta_1 = 0.0,delta_2 = 0.0
var = 3, beta_1 = Inf, beta_2 = 0.0, delta_1 = 0.0,delta_2 = 0.0
var = 4, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 5, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 6, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 7, beta_1 = -0.0, beta_2 = 500.0, delta_1 = 2.0,delta_2 = 2.0
Sensitivity for coefficients in objective:
var = 1, beta_1 = Inf, beta_2 = 3.0, delta_1 = 300.0,delta_2 = 300.0
var = 2, beta_1 = Inf, beta_2 = Inf, delta_1 = 100.0,delta_2 = 100.0
var = 3, beta_1 = -2.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0
var = 4, beta_1 = Inf, beta_2 = 2.0, delta_1 = 500.0,delta_2 = 500.0
var = 5, beta_1 = -3.0, beta_2 = Inf, delta_1 = 500.0,delta_2 = 500.0
var = 6, beta_1 = Inf, beta_2 = 2.0, delta_1 = 500.0,delta_2 = 500.0
var = 7, beta_1 = -2.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0


Optimal partition sensitivity results:

Sensitivity for bounds on constraints:
con = 1, beta_1 = -300.0, beta_2 = 500.0, delta_1 = 3.0,delta_2 = 1.0
con = 2, beta_1 = -700.0, beta_2 = Inf, delta_1 = -0.0,delta_2 = -0.0
con = 3, beta_1 = -500.0, beta_2 = 500.0, delta_1 = 3.0,delta_2 = 1.0
con = 4, beta_1 = -500.0, beta_2 = 500.0, delta_1 = 2.0,delta_2 = 4.0
con = 5, beta_1 = -100.0, beta_2 = 300.0, delta_1 = 3.0,delta_2 = 5.0
con = 6, beta_1 = -500.0, beta_2 = 700.0, delta_1 = 3.0,delta_2 = 5.0
con = 7, beta_1 = -500.0, beta_2 = 700.0, delta_1 = 2.0,delta_2 = 2.0
```

```
Sensitivity for bounds on variables:
var = 1, beta_1 = Inf, beta_2 = 300.0, delta_1 = 0.0,delta_2 = 0.0
var = 2, beta_1 = Inf, beta_2 = 100.0, delta_1 = 0.0,delta_2 = 0.0
var = 3, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 2.0
var = 4, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 5, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 6, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 7, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 2.0
Sensitivity for coefficients in objective:
var = 1, beta_1 = Inf, beta_2 = 3.0, delta_1 = 300.0,delta_2 = 300.0
var = 2, beta_1 = Inf, beta_2 = Inf, delta_1 = 100.0,delta_2 = 100.0
var = 3, beta_1 = -2.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0
var = 4, beta_1 = Inf, beta_2 = 2.0, delta_1 = 500.0,delta_2 = 500.0
var = 5, beta_1 = -3.0, beta_2 = Inf, delta_1 = 500.0,delta_2 = 500.0
var = 6, beta_1 = Inf, beta_2 = 2.0, delta_1 = 500.0,delta_2 = 500.0
var = 7, beta_1 = -2.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0
```

# Chapter 17

# 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.

## 17.1 Linear optimization

A linear optimization problem can be written as

$$
\begin{array}{llcccl}
\text{minimize} & & & c^T x + c^f & & \\
\text{subject to} & l^c & \leq & Ax & \leq & u^c, \\
& l^x & \leq & x & \leq & u^x,
\end{array}
\tag{17.1}
$$

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 (17.1). If (17.1) has at least one primal feasible solution, then (17.1) is said to be (primal) feasible.

In case (17.1) does not have a feasible solution, the problem is said to be *(primal) infeasible* .

## 17.1.1    Duality for linear optimization

Corresponding to the primal problem (17.1), there is a dual problem

$$
\begin{array}{rrcl}
\text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f \\
\text{subject to} & A^T y + s_l^x - s_u^x &=& c, \\
& -y + s_l^c - s_u^c &=& 0, \\
& s_l^c, s_u^c, s_l^x, s_u^x &\geq& 0.
\end{array}
\tag{17.2}
$$

If a bound in the primal problem is plus or minus infinity, the corresponding dual variable is fixed at 0, and we use the convention that the product of the bound value and the corresponding dual variable is 0. E.g.

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

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

A solution

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

to the dual problem is feasible if it satisfies all the constraints in (17.2). If (17.2) has at least one feasible solution, then (17.2) is *(dual) feasible*, otherwise the problem is *(dual) infeasible*.

### 17.1.1.1    A primal-dual feasible solution

A solution

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

is denoted a *primal-dual feasible solution*, if $(x)$ is a solution to the primal problem (17.1) and $(y, s_l^c, s_u^c, s_l^x, s_u^x)$ is a solution to the corresponding dual problem (17.2).

### 17.1.1.2    The duality gap

Let

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

be a primal-dual feasible solution, and let

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

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

$$
\begin{aligned}
& c^T x^* + c^f - \left( (l^c)^T (s_l^c)^* - (u^c)^T (s_u^c)^* + (l^x)^T (s_l^x)^* - (u^x)^T (s_u^x)^* + c^f \right) \\
&= \sum_{i=0}^{m-1} \left[ (s_l^c)_i^* ((x_i^c)^* - l_i^c) + (s_u^c)_i^* (u_i^c - (x_i^c)^*) \right] + \sum_{j=0}^{n-1} \left[ (s_l^x)_j^* (x_j - l_j^x) + (s_u^x)_j^* (u_j^x - x_j^*) \right] \quad (17.3) \\
& \geq 0
\end{aligned}
$$

where the first relation can be obtained by transposing and multiplying the dual constraints (17.2) 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.

### 17.1.1.3 When the objective is to be maximized

When the objective sense of problem (17.1) is maximization, i.e.

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

the objective sense of the dual problem changes to minimization, and the domain of all dual variables changes sign in comparison to (17.2). The dual problem thus takes the form

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

This means that the duality gap, defined in (17.3) 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.

### 17.1.1.4 An optimal solution

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

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

are satisfied.

If (17.1) 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.

## 17.1.2   Infeasibility for linear optimization

### 17.1.2.1   Primal infeasible problems

If the problem (17.1) 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

$$\begin{array}{llrcl}
\text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x \\
\text{subject to} & A^T y + s_l^x - s_u^x & = & 0, \\
& -y + s_l^c - s_u^c & = & 0, \\
& s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0,
\end{array} \tag{17.4}$$

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 (17.4) 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 (17.4) is unbounded, and that its dual is infeasible. As the constraints to the dual of (17.4) is identical to the constraints of problem (17.1), we thus have that problem (17.1) is also infeasible.

### 17.1.2.2   Dual infeasible problems

If the problem (17.2) 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

$$\begin{array}{llccccc}
\text{minimize} & & & c^T x \\
\text{subject to} & \hat{l}^c & \leq & Ax & \leq & \hat{u}^c, \\
& \hat{l}^x & \leq & x & \leq & \hat{u}^x,
\end{array} \tag{17.5}$$

where

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

and

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

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

Such a solution implies that (17.5) is unbounded, and that its dual is infeasible. As the constraints to the dual of (17.5) is identical to the constraints of problem (17.2), we thus have that problem (17.2) is also infeasible.

#### 17.1.2.3 Primal and dual infeasible case

In case that both the primal problem (17.1) and the dual problem (17.2) are infeasible, MOSEK will report only one of the two possible certificates — which one is not defined (MOSEK returns the first certificate found).

## 17.2 Conic quadratic optimization

*Conic quadratic optimization* is an extensions of linear optimization (see Section 17.1) allowing conic domains to be specified for subsets of the problem variables. A conic quadratic optimization problem can be written as

$$\begin{array}{lllll} \text{minimize} & & c^T x + c^f & & \\ \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \\ & & x \in \mathcal{C}, & & \end{array} \tag{17.6}$$

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

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

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

MOSEK supports only a limited number of cones, specifically:

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

- The quadratic cone:

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

- The rotated quadratic cone:

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

Although these cones may seem to provide only limited expressive power they can be used to model a wide range of problems as demonstrated in [20].

## 17.2.1   Duality for conic quadratic optimization

The dual problem corresponding to the conic quadratic optimization problem (17.6) is given by

$$
\begin{array}{lrcl}
\text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f \\
\text{subject to} & A^T y + s_l^x - s_u^x + s_n^x & = & c, \\
& -y + s_l^c - s_u^c & = & 0, \\
& s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0, \\
& s_n^x \in \mathcal{C}^*,
\end{array}
\tag{17.7}
$$

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

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

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

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

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

- The quadratic cone:

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

- The rotated quadratic cone:

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

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

## 17.2.2  Infeasibility for conic quadratic optimization

In case MOSEK finds a problem to be infeasible it reports a certificate of the infeasibility. This works exactly as for linear problems (see Section 17.1.2).

### 17.2.2.1  Primal infeasible problems

If the problem (17.6) is infeasible, MOSEK will report a certificate of primal infeasibility: The dual solution reported is the certificate of infeasibility, and the primal solution is undefined.

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

$$
\begin{array}{lrcl}
\text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x & & \\
\text{subject to} & A^T y + s_l^x - s_u^x + s_n^x & = & 0, \\
& -y + s_l^c - s_u^c & = & 0, \\
& s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0, \\
& s_n^x \in \mathcal{C}^*, & &
\end{array}
\tag{17.8}
$$

such that the objective value is strictly positive.

### 17.2.2.2  Dual infeasible problems

If the problem (17.7) is infeasible, MOSEK will report a certificate of dual infeasibility: The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

$$
\begin{array}{lrcccl}
\text{minimize} & & & c^T x & & \\
\text{subject to} & \hat{l}^c & \leq & Ax & \leq & \hat{u}^c, \\
& \hat{l}^x & \leq & x & \leq & \hat{u}^x, \\
& & & x \in \mathcal{C}, & &
\end{array}
\tag{17.9}
$$

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.

## 17.3    Semidefinite optimization

*Semidefinite optimization* is an extension of conic quadratic optimization (see Section 17.2) allowing positive semidefinite matrix variables to be used in addition to the usual scalar variables. A semidefinite optimization problem can be written as

$$
\begin{array}{llrcll}
\text{minimize} & & \displaystyle\sum_{j=0}^{n-1} c_j x_j + \sum_{j=0}^{p-1} \left\langle \overline{C}_j, \overline{X}_j \right\rangle + c^f & & & \\[2ex]
\text{subject to} & l_i^c & \leq \quad \displaystyle\sum_{j=0}^{n-1} a_{ij} x_j + \sum_{j=0}^{p-1} \left\langle \overline{A}_{ij}, \overline{X}_j \right\rangle & \leq & u_i^c, & i = 0, \ldots, m-1 \\[2ex]
& l_j^x & \leq \qquad\qquad x_j & \leq & u_j^x, & j = 0, \ldots, n-1 \\[1ex]
& & x \in \mathcal{C}, \overline{X}_j \in \mathcal{S}_{r_j}^+, & & & j = 0, \ldots, p-1
\end{array}
\tag{17.10}
$$

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

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

With semidefinite optimization we can model a wide range of problems as demonstrated in [20].

### 17.3.1    Duality for semidefinite optimization

The dual problem corresponding to the semidefinite optimization problem (17.10) is given by

$$
\begin{array}{llrcll}
\text{maximize} & & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f & & & \\[1ex]
\text{subject to} & c - A^T y + s_u^x - s_l^x & = & s_n^x, & & \\[1ex]
& \overline{C}_j - \displaystyle\sum_{i=0}^{m} y_i \overline{A}_{ij} & = & \overline{S}_j, & j = 0, \ldots, p-1 \\[2ex]
& s_l^c - s_u^c & = & y, & & \\[1ex]
& s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0, & & \\[1ex]
& s_n^x \in \mathcal{C}^*, \ \overline{S}_j \in \mathcal{S}_{r_j}^+, & & & j = 0, \ldots, p-1
\end{array}
\tag{17.11}
$$

where $A \in \mathbb{R}^{m \times n}$, $A_{ij} = a_{ij}$, which is similar to the dual problem for conic quadratic optimization (see Section 17.7), 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.

## 17.3.2 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 17.1.2).

### 17.3.2.1 Primal infeasible problems

If the problem (17.10) is infeasible, MOSEK will report a certificate of primal infeasibility: The dual solution reported is a certificate of infeasibility, and the primal solution is undefined.

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

$$
\begin{aligned}
\text{maximize} \qquad & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x \\
\text{subject to} \qquad & A^T y + s_l^x - s_u^x + s_n^x = 0, \\
& \sum_{i=0}^{m-1} y_i \overline{A}_{ij} + \overline{S}_j = 0, \qquad\qquad j = 0, \ldots, p-1 \\
& -y + s_l^c - s_u^c = 0, \\
& s_l^c, s_u^c, s_l^x, s_u^x \geq 0, \\
& s_n^x \in \mathcal{C}^*, \ \overline{S}_j \in \mathcal{S}_{r_j}^+, \qquad j = 0, \ldots, p-1
\end{aligned}
\tag{17.12}
$$

such that the objective value is strictly positive.

### 17.3.2.2 Dual infeasible problems

If the problem (17.11) is infeasible, MOSEK will report a certificate of dual infeasibility: The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

$$
\begin{aligned}
\text{minimize} \qquad & \sum_{j=0}^{n-1} c_j x_j + \sum_{j=0}^{p-1} \left\langle \overline{C}_j, \overline{X}_j \right\rangle \\
\text{subject to} \quad \hat{l}_i^c \leq & \sum_{j=1}^{p-1} a_{ij} x_j + \sum_{j=0}^{p-1} \left\langle \overline{A}_{ij}, \overline{X}_j \right\rangle \leq \hat{u}_i^c, \quad i = 0, \ldots, m-1 \\
\hat{l}^x \leq & \qquad\qquad x \qquad\qquad \leq \hat{u}^x, \\
& x \in \mathcal{C}, \ \overline{X}_j \in \mathcal{S}_{r_j}^+, \qquad\qquad j = 0, \ldots, p-1
\end{aligned}
\tag{17.13}
$$

where

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

and

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

such that the objective value is strictly negative.

## 17.4   Quadratic and quadratically constrained optimization

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

$$
\begin{array}{llcccl}
\text{minimize} & & & \frac{1}{2}x^T Q^o x + c^T x + c^f & & \\
\text{subject to} & l_k^c & \leq & \frac{1}{2}x^T Q^k x + \sum_{j=0}^{n-1} a_{kj}x_j & \leq & u_k^c, \quad k = 0,\dots,m-1, \\
& l_j^x & \leq & x_j & \leq & u_j^x, \quad j = 0,\dots,n-1,
\end{array}
\tag{17.14}
$$

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

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

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

Note that any convex quadratic and quadratically constrained optimization problem can be reformulated as a conic optimization problem. It is our experience that for the majority of practical applications it is better to cast them as conic problems because

- the resulting problem is convex by construction, and

- the conic optimizer is more efficient than the optimizer for general quadratic problems.

See [20] for further details.

### 17.4.1   Duality for quadratic and quadratically constrained optimization

The dual problem corresponding to the quadratic and quadratically constrained optimization problem (17.14) is given by

$$
\begin{array}{lrcl}
\text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + \frac{1}{2}x^T \left( \sum_{k=0}^{m-1} y_k Q^k - Q^o \right) x + c^f & & \\
\text{subject to} & A^T y + s_l^x - s_u^x + \left( \sum_{k=0}^{m-1} y_k Q^k - Q^o \right) x & = & c, \\
& -y + s_l^c - s_u^c & = & 0, \\
& s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0.
\end{array}
\tag{17.15}
$$

The dual problem is related to the dual problem for linear optimization (see Section 17.2), but depend on variable $x$ which in general can not be eliminated. In the solutions reported by MOSEK, the value of $x$ is the same for the primal problem (17.14) and the dual problem (17.15).

### 17.4.2 Infeasibility for quadratic and quadratically constrained optimization

In case MOSEK finds a problem to be infeasible it reports a certificate of the infeasibility. This works exactly as for linear problems (see Section 17.1.2).

#### 17.4.2.1 Primal infeasible problems

If the problem (17.14) with all $Q^k = 0$ is infeasible, MOSEK will report a certificate of primal infeasibility. As the constraints is the same as for a linear problem, the certificate of infeasibility is the same as for linear optimization (see Section 17.1.2.1).

#### 17.4.2.2 Dual infeasible problems

If the problem (17.15) with all $Q^k = 0$ is infeasible, MOSEK will report a certificate of dual infeasibility: The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

$$
\begin{array}{rlrcl}
\text{minimize} & & & c^T x \\
\text{subject to} & \hat{l}^c & \leq & Ax & \leq & \hat{u}^c, \\
& 0 & \leq & Q^o x & \leq & 0, \\
& \hat{l}^x & \leq & x & \leq & \hat{u}^x,
\end{array}
\tag{17.16}
$$

where

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

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.

## 17.5 General convex optimization

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

$$
\begin{array}{llccl}
\text{minimize} & & f(x) + c^T x + c^f & & \\
\text{subject to} & l^c \leq & g(x) + Ax & \leq & u^c, \\
& l^x \leq & x & \leq & u^x,
\end{array}
\tag{17.17}
$$

where

- $m$ is the number of constraints.

- $n$ is the number of decision variables.

- $x \in \mathbb{R}^n$ is a vector of decision variables.

- $c \in \mathbb{R}^n$ is the linear part objective function.

- $A \in \mathbb{R}^{m \times n}$ is the constraint matrix.

- $l^c \in \mathbb{R}^m$ is the lower limit on the activity for the constraints.

- $u^c \in \mathbb{R}^m$ is the upper limit on the activity for the constraints.

- $l^x \in \mathbb{R}^n$ is the lower limit on the activity for the variables.

- $u^x \in \mathbb{R}^n$ is the upper limit on the activity for the variables.

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

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

This means that the $i$th constraint has the form

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

The linear term $Ax$ is not included in $g(x)$ since it can be handled much more efficiently as a separate entity when optimizing.

The nonlinear functions $f$ and $g$ must be smooth in all $x \in [l^x; u^x]$. Moreover, $f(x)$ must be a convex function and $g_i(x)$ must satisfy

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

## 17.5.1   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

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

and the dual problem is given by

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

which is equivalent to

$$
\begin{array}{rll}
\text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f & \\
& + f(x) - g(x)^T y - (\nabla f(x)^T - \nabla g(x)^T y)^T x & \\
\text{subject to} & A^T y + s_l^x - s_u^x - (\nabla f(x)^T - \nabla g(x)^T y) & = \quad c, \\
& - y + s_l^c - s_u^c & = \quad 0, \\
& s_l^c, s_u^c, s_l^x, s_u^x \geq 0. &
\end{array} \qquad (17.18)
$$

In this context we use the following definition for scalar functions

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

and accordingly for vector functions

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

# Chapter 18

# 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 [2].

## 18.1  MPS file structure

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

$$
\begin{array}{rcccl}
l^c & \leq & Ax + q(x) & \leq & u^c, \\
l^x & \leq & x & \leq & u^x, \\
& & x \in \mathcal{C}, \\
& & x_{\mathcal{J}} \text{ integer},
\end{array}
\tag{18.1}
$$

where

- $x \in \mathbb{R}^n$ is the vector of decision variables.

- $A \in \mathbb{R}^{m \times n}$ is the constraint matrix.

- $l^c \in \mathbb{R}^m$ is the lower limit on the activity for the constraints.

- $u^c \in \mathbb{R}^m$ is the upper limit on the activity for the constraints.

- $l^x \in \mathbb{R}^n$ is the lower limit on the activity for the variables.

- $u^x \in \mathbb{R}^n$ is the upper limit on the activity for the variables.

- $q : \mathbb{R}^n \to \mathbb{R}$ is a vector of quadratic functions. Hence,

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

  where it is assumed that

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

Please note the explicit $1/2$ in the quadratic term and that $Q^i$ is required to be symmetric.

- $\mathcal{C}$ is a convex cone.

- $\mathcal{J} \subseteq \{1, 2, \ldots, n\}$ is an index set of the integer-constrained variables.

An MPS file with one row and one column can be illustrated like this:

```
*         1         2         3         4         5         6
*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]
BOUNDS
 ?? [name]    [vname1]    [value1]
CSECTION        [kname1]    [value1]      [ktype]
    [vname1]
ENDATA
```

Here the names in capitals are keywords of the MPS format and names in brackets are custom defined names or values. A couple of notes on the structure:

Fields:

All items surrounded by brackets appear in *fields*. The fields named "valueN" are numerical values. Hence, they must have the format

`[+|-]XXXXXXX.XXXXXX[[e|E][+|-]XXX]`

where

`X = [0|1|2|3|4|5|6|7|8|9].`

Sections:

The MPS file consists of several sections where the names in capitals indicate the beginning of a new section. For example, `COLUMNS` denotes the beginning of the columns section.

Comments:

Lines starting with an "*" are comment lines and are ignored by MOSEK.

Keys:

The question marks represent keys to be specified later.

Extensions:

The sections `QSECTION` and `CSECTION` are MOSEK specific extensions of the MPS format.

The standard MPS format is a fixed format, i.e. everything in the MPS file must be within certain fixed positions. MOSEK also supports a *free format*. See Section 18.5 for details.

### 18.1.1 Linear example `lo1.mps`

A concrete example of a MPS file is presented below:

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

Subsequently each individual section in the MPS format is discussed.

### 18.1.2 `NAME`

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

### 18.1.3 `OBJSENSE` (optional)

This is an optional section that can be used to specify the sense of the objective function. The `OBJSENSE` section contains one line at most which can be one of the following

```
MIN
MINIMIZE
MAX
MAXIMIZE
```

It should be obvious what the implication is of each of these four lines.


### 18.1.4   OBJNAME (optional)

This is an optional section that can be used to specify the name of the row that is used as objective function. The OBJNAME section contains one line at most which has the form

```
objname
```

objname should be a valid row name.


### 18.1.5   ROWS

A record in the ROWS section has the form

```
?   [cname1]
```

where the requirements for the fields are as follows:

| Field | Starting position | Maximum width | Re-quired | Description |
|---|---|---|---|---|
| ? | 2 | 1 | Yes | Constraint key |
| [cname1] | 5 | 8 | Yes | Constraint name |

Hence, in this section each constraint is assigned an unique name denoted by [cname1]. Please note that [cname1] starts in position 5 and the field can be at most 8 characters wide. An initial key (?) must be present to specify the type of the constraint. The key can have the values E, G, L, or N with the following interpretation:

| Constraint type | $l_i^c$ | $u_i^c$ |
|---|---|---|
| E | finite | $l_i^c$ |
| G | finite | $\infty$ |
| L | $-\infty$ | finite |
| N | $-\infty$ | $\infty$ |

In the MPS format an objective vector is not specified explicitly, but one of the constraints having the key N will be used as the objective vector $c$. In general, if multiple N type constraints are specified, then the first will be used as the objective vector $c$.


### 18.1.6   COLUMNS

In this section the elements of $A$ are specified using one or more records having the form

```
[vname1]  [cname1]    [value1]     [cname2]  [value2]
```

where the requirements for each field are as follows:

| Field | Starting position | Maximum width | Re-quired | Description |
|---|---|---|---|---|
| [vname1] | 5 | 8 | Yes | Variable name |
| [cname1] | 15 | 8 | Yes | Constraint name |
| [value1] | 25 | 12 | Yes | Numerical value |
| [cname2] | 40 | 8 | No | Constraint name |
| [value2] | 50 | 12 | No | Numerical value |

Hence, a record specifies one or two elements $a_{ij}$ of $A$ using the principle that [vname1] and [cname1] determines $j$ and $i$ respectively. Please note that [cname1] must be a constraint name specified in the ROWS section. Finally, [value1] denotes the numerical value of $a_{ij}$ . Another optional element is specified by [cname2], and [value2] for the variable specified by [vname1]. Some important comments are:

- All elements belonging to one variable must be grouped together.

- Zero elements of $A$ should not be specified.

- At least one element for each variable should be specified.

## 18.1.7 RHS (optional)

A record in this section has the format

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

where the requirements for each field are as follows:

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

The interpretation of a record is that [name] is the name of the RHS vector to be specified. In general, several vectors can be specified. [cname1] denotes a constraint name previously specified in the ROWS section. Now, assume that this name has been assigned to the $i$ th constraint and $v_1$ denotes the value specified by [value1], then the interpretation of $v_1$ is:

| Constraint type | $l_i^c$ | $u_i^c$ |
|---|---|---|
| E | $v_1$ | $v_1$ |
| G | $v_1$ | |
| L | | $v_1$ |
| N | | |

An optional second element is specified by [cname2] and [value2] and is interpreted in the same way. Please note that it is not necessary to specify zero elements, because elements are assumed to be zero.

## 18.1.8   RANGES (optional)

A record in this section has the form

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

where the requirements for each fields are as follows:

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

The records in this section are used to modify the bound vectors for the constraints, i.e. the values in $l^c$ and $u^c$ . A record has the following interpretation: [name] is the name of the RANGE vector 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 | | | |

## 18.1.9   QSECTION (optional)

Within the QSECTION the label [cname1] must be a constraint name previously specified in the ROWS section. The label [cname1] denotes the constraint to which the quadratic term belongs. A record in the QSECTION has the form

 [vname1]   [vname2]     [value1]      [vname3]   [value2]

where the requirements for each field are:

| Field | Starting position | Maximum width | Re- quired | Description |
|---|---|---|---|---|
| [vname1] | 5 | 8 | Yes | Variable name |
| [vname2] | 15 | 8 | Yes | Variable name |
| [value1] | 25 | 12 | Yes | Numerical value |
| [vname3] | 40 | 8 | No | Variable name |
| [value2] | 50 | 12 | No | Numerical value |

A record specifies one or two elements in the lower triangular part of the $Q^i$ matrix where [cname1] specifies the $i$ . Hence, if the names [vname1] and [vname2] have been assigned to the $k$ th and $j$ th variable, then $Q^i_{kj}$ is assigned the value given by [value1] An optional second element is specified in the same way by the fields [vname1], [vname3], and [value2].

The example

$$\begin{array}{lcr} \text{minimize} & -x_2 + 0.5(2x_1^2 - 2x_1x_3 + 0.2x_2^2 + 2x_3^2) & \\ \text{subject to} & x_1 + x_2 + x_3 & \geq \quad 1, \\ & x \geq 0 & \end{array}$$

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 QSECTIONs please note that:

- Only one QSECTION is allowed for each constraint.

- The QSECTIONs can appear in an arbitrary order after the COLUMNS section.

- All variable names occurring in the QSECTION must already be specified in the COLUMNS section.

- All entries specified in a QSECTION are assumed to belong to the lower triangular part of the quadratic term of $Q$ .

## 18.1.10   BOUNDS (optional)

In the BOUNDS section changes to the default bounds vectors $l^x$ and $u^x$ are specified. The default bounds vectors are $l^x = 0$ and $u^x = \infty$ . Moreover, it is possible to specify several sets of bound vectors. A record in this section has the form

```
?? [name]     [vname1]     [value1]
```

where the requirements for each field are:

| Field | Starting position | Maximum width | Re- quired | Description |
|---|---|---|---|---|
| ?? | 2 | 2 | Yes | Bound key |
| [name] | 5 | 8 | Yes | Name of the BOUNDS vector |
| [vname1] | 15 | 8 | Yes | Variable name |
| [value1] | 25 | 12 | No | Numerical value |

Hence, a record in the BOUNDS section has the following interpretation: [name] is the name of the bound vector and [vname1] is the name of the variable which bounds are modified by the record. ?? and [value1] are used to modify the bound vectors according to the following table:

| ?? | $l_j^x$ | $u_j^x$ | Made integer (added to $\mathcal{J}$ ) |
|---|---|---|---|
| FR | $-\infty$ | $\infty$ | No |
| FX | $v_1$ | $v_1$ | No |
| LO | $v_1$ | unchanged | No |
| MI | $-\infty$ | unchanged | No |
| PL | unchanged | $\infty$ | No |
| UP | unchanged | $v_1$ | No |
| BV | 0 | 1 | Yes |
| LI | $\lceil v_1 \rceil$ | *unchanged* | Yes |
| UI | unchanged | $\lfloor v_1 \rfloor$ | Yes |

$v_1$ is the value specified by [value1].

## 18.1.11   CSECTION (optional)

The purpose of the CSECTION is to specify the constraint

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

in (18.1).

It is assumed that $\mathcal{C}$ satisfies the following requirements. Let

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

be vectors comprised of parts of the decision variables $x$ so that each decision variable is a member of exactly **one** vector $x^t$ , for example

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

Next define

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

where $\mathcal{C}_t$ must have one of the following forms

- $\mathbb{R}$ set:

$$\mathcal{C}_t = \{ x \in \mathbb{R}^{n^t} \}.$$

- Quadratic cone:

$$\mathcal{C}_t = \left\{ x \in \mathbb{R}^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\}. \tag{18.2}$$

- Rotated quadratic cone:

$$\mathcal{C}_t = \left\{ x \in \mathbb{R}^{n^t} : 2x_1 x_2 \geq \sum_{j=3}^{n^t} x_j^2, \ x_1, x_2 \geq 0 \right\}. \tag{18.3}$$

In general, only quadratic and rotated quadratic cones are specified in the MPS file whereas membership of the $\mathbb{R}$ set is not. If a variable is not a member of any other cone then it is assumed to be a member of an $\mathbb{R}$ cone.

Next, let us study an example. Assume that the quadratic cone

$$x_4 \geq \sqrt{x_5^2 + x_8^2} \tag{18.4}$$

and the rotated quadratic cone

$$2x_3 x_7 \geq x_1^2 + x_0^2, \ x_3, x_7 \geq 0, \tag{18.5}$$

should be specified in the MPS file. One CSECTION is required for each cone and they are specified as follows:

```
*         1         2         3         4         5         6
*234567890123456789012345678901234567890123456789012345678901234567890
 CSECTION      konea       0.0           QUAD
    x4
    x5
    x8
```

```
CSECTION        koneb        0.0          RQUAD
     x7
     x3
     x1
     x0
```

This first `CSECTION` specifies the cone (18.4) which is given the name `konea`. This is a quadratic cone which is specified by the keyword `QUAD` in the `CSECTION` header. The 0.0 value in the `CSECTION` header is not used by the `QUAD` cone.

The second `CSECTION` specifies the rotated quadratic cone (18.5). Please note the keyword `RQUAD` in the `CSECTION` which is used to specify that the cone is a rotated quadratic cone instead of a quadratic cone. The 0.0 value in the `CSECTION` header is not used by the `RQUAD` cone.

In general, a `CSECTION` header has the format

```
 CSECTION      [kname1]     [value1]      [ktype]
```

where the requirement for each field are as follows:

| Field | Starting position | Maximum width | Re-quired | 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 | $\geq 1$ | Quadratic cone i.e. (18.2). |
| RQUAD | $\geq 2$ | Rotated quadratic cone i.e. (18.3). |

Please note that a quadratic cone must have at least one member whereas a rotated quadratic cone must have at least two members. A record in the `CSECTION` has the format

```
 [vname1]
```

where the requirements for each field are

| Field | Starting position | Maximum width | Re-quired | Description |
|---|---|---|---|---|
| [vname1] | 2 | 8 | Yes | A valid variable name |

The most important restriction with respect to the `CSECTION` is that a variable must occur in only one `CSECTION`.

### 18.1.12   ENDATA

This keyword denotes the end of the MPS file.

## 18.2  Integer variables

Using special bound keys in the `BOUNDS` section it is possible to specify that some or all of the variables should be integer-constrained i.e. be members of $\mathcal{J}$ . However, an alternative method is available.

This method is available only for backward compatibility and we recommend that it is not used. This method requires that markers are placed in the `COLUMNS` section as in the example:

```
COLUMNS
    x1        obj       -10.0         c1        0.7
    x1        c2        0.5           c3        1.0
    x1        c4        0.1
* Start of integer-constrained variables.
    MARK000   'MARKER'                'INTORG'
    x2        obj       -9.0          c1        1.0
    x2        c2        0.8333333333  c3        0.66666667
    x2        c4        0.25
    x3        obj       1.0           c6        2.0
    MARK001   'MARKER'                'INTEND'
* End of integer-constrained variables.
```

Please note that special marker lines are used to indicate the start and the end of the integer variables. Furthermore be aware of the following

- IMPORTANT: All variables between the markers are assigned a default lower bound of 0 and a default upper bound of 1. **This may not be what is intended.** If it is not intended, the correct bounds should be defined in the `BOUNDS` section of the MPS formatted file.

- MOSEK ignores field 1, i.e. `MARK0001` and `MARK001`, however, other optimization systems require them.

- Field 2, i.e. `'MARKER'`, must be specified including the single quotes. This implies that no row can be assigned the name `'MARKER'`.

- Field 3 is ignored and should be left blank.

- Field 4, i.e. `'INTORG'` and `'INTEND'`, must be specified.

- It is possible to specify several such integer marker sections within the `COLUMNS` section.

## 18.3  General limitations

- An MPS file should be an ASCII file.

## 18.4  Interpretation of the MPS format

Several issues related to the MPS format are not well-defined by the industry standard. However, MOSEK uses the following interpretation:

- If a matrix element in the COLUMNS section is specified multiple times, then the multiple entries are added together.

- If a matrix element in a QSECTION section is specified multiple times, then the multiple entries are added together.

## 18.5   The free MPS format

MOSEK supports a free format variation of the MPS format. The free format is similar to the MPS file format but less restrictive, e.g. it allows longer names. However, it also presents two main limitations:

- By default a line in the MPS file must not contain more than 1024 characters. However, by modifying the parameter MSK_IPAR_READ_MPS_WIDTH an arbitrary large line width will be accepted.

- A name must not contain any blanks.

To use the free MPS format instead of the default MPS format the MOSEK parameter MSK_IPAR_READ_MPS_FORMAT should be changed.

# Chapter 19

# The LP file format

MOSEK supports the LP file format with some extensions i.e. MOSEK can read and write LP formatted files.

Please note that the LP format is not a completely well-defined standard and hence different optimization packages may interpret the same LP file in slightly different ways. MOSEK tries to emulate as closely as possible CPLEX's behavior, but tries to stay backward compatible.

The LP file format can specify problems on the form

$$
\begin{array}{llrcl}
\text{minimize/maximize} & & \multicolumn{3}{c}{c^T x + \dfrac{1}{2} q^o(x)} \\
\text{subject to} & l^c & \leq & Ax + \dfrac{1}{2} q(x) & \leq & u^c, \\
& l^x & \leq & x & \leq & u^x, \\
& & & x_{\mathcal{J}} \text{integer,} &
\end{array}
$$

where

- $x \in \mathbb{R}^n$ is the vector of decision variables.

- $c \in \mathbb{R}^n$ is the linear term in the objective.

- $q^o :\in \mathbb{R}^n \to \mathbb{R}$ is the quadratic term in the objective where

$$
q^o(x) = x^T Q^o x
$$

  and it is assumed that

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

- $A \in \mathbb{R}^{m \times n}$ is the constraint matrix.

- $l^c \in \mathbb{R}^m$ is the lower limit on the activity for the constraints.

- $u^c \in \mathbb{R}^m$ is the upper limit on the activity for the constraints.

- $l^x \in \mathbb{R}^n$ is the lower limit on the activity for the variables.

- $u^x \in \mathbb{R}^n$ is the upper limit on the activity for the variables.

- $q : \mathbb{R}^n \to \mathbb{R}$ is a vector of quadratic functions. Hence,

$$q_i(x) = x^T Q^i x$$

  where it is assumed that

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

- $\mathcal{J} \subseteq \{1, 2, \ldots, n\}$ is an index set of the integer constrained variables.

## 19.1   The sections

An LP formatted file contains a number of sections specifying the objective, constraints, variable bounds, and variable types. The section keywords may be any mix of upper and lower case letters.

### 19.1.1   The objective

The first section beginning with one of the keywords

```
max
maximum
maximize
min
minimum
minimize
```

defines the objective sense and the objective function, i.e.

$$c^T x + \frac{1}{2} x^T Q^o x.$$

The objective may be given a name by writing

```
myname:
```

before the expressions. If no name is given, then the objective is named `obj`.

The objective function contains linear and quadratic terms. The linear terms are written as

```
4 x1 + x2 - 0.1 x3
```

and so forth. The quadratic terms are written in square brackets (`[]`) and are either squared or multiplied as in the examples

```
x1^2
```

and

```
x1 * x2
```

There may be zero or more pairs of brackets containing quadratic expressions.

An example of an objective section is:

```
minimize
myobj: 4 x1 + x2 - 0.1 x3 + [ x1^2 + 2.1 x1 * x2 ]/2
```

Please note that the quadratic expressions are multiplied with $\frac{1}{2}$ , so that the above expression means

$$\text{minimize} \quad 4x_1 + x_2 - 0.1 \cdot x_3 + \frac{1}{2}(x_1^2 + 2.1 \cdot x_1 \cdot x_2)$$

If the same variable occurs more than once in the linear part, the coefficients are added, so that `4 x1 + 2 x1` is equivalent to `6 x1`. In the quadratic expressions `x1 * x2` is equivalent to `x2 * x1` and as in the linear part , if the same variables multiplied or squared occur several times their coefficients are added.

## 19.1.2 The constraints

The second section beginning with one of the keywords

```
subj to
subject to
s.t.
st
```

defines the linear constraint matrix ($A$ ) and the quadratic matrices ($Q^i$ ).

A constraint contains a name (optional), expressions adhering to the same rules as in the objective and a bound:

```
subject to
con1: x1 + x2 + [ x3^2 ]/2 <= 5.1
```

The bound type (here `<=`) may be any of `<, <=, =, >, >=` (`<` and `<=` mean the same), and the bound may be any number.

In the standard LP format it is not possible to define more than one bound, but MOSEK supports defining ranged constraints by using double-colon (`''::''`) instead of a single-colon (`":"`) after the constraint name, i.e.

$$-5 \leq x_1 + x_2 \leq 5 \tag{19.1}$$

may be written as

```
con:: -5 < x_1 + x_2 < 5
```

By default MOSEK writes ranged constraints this way.

If the files must adhere to the LP standard, ranged constraints must either be split into upper bounded and lower bounded constraints or be written as en equality with a slack variable. For example the expression (19.1) may be written as

$$x_1 + x_2 - sl_1 = 0, -5 \leq sl_1 \leq 5.$$

### 19.1.3    Bounds

Bounds on the variables can be specified in the bound section beginning with one of the keywords

```
bound
bounds
```

The bounds section is optional but should, if present, follow the `subject to` section. All variables listed in the bounds section must occur in either the objective or a constraint.

The default lower and upper bounds are 0 and $+\infty$ . A variable may be declared free with the keyword `free`, which means that the lower bound is $-\infty$ and the upper bound is $+\infty$ . Furthermore it may be assigned a finite lower and upper bound. The bound definitions for a given variable may be written in one or two lines, and bounds can be any number or $\pm\infty$ (written as `+inf/-inf/+infinity/-infinity`) as in the example

```
bounds
x1 free
x2 <= 5
0.1 <= x2
x3 = 42
2 <= x4 < +inf
```

### 19.1.4    Variable types

The final two sections are optional and must begin with one of the keywords

```
bin
binaries
binary
```

and

```
gen
general
```

Under `general` all integer variables are listed, and under `binary` all binary (integer variables with bounds 0 and 1) are listed:

```
general
x1 x2
binary
x3 x4
```

Again, all variables listed in the binary or general sections must occur in either the objective or a constraint.

### 19.1.5    Terminating section

Finally, an LP formatted file must be terminated with the keyword

```
end
```

### 19.1.6  Linear example `lo1.lp`

A simple example of an LP file is:

```
\ File: lo1.lp
maximize
obj: 3 x1 + x2 + 5 x3 + x4
subject to
c1:  3 x1 + x2 + 2 x3 = 30
c2:  2 x1 + x2 + 3 x3 + x4 >= 15
c3:  2 x2 + 3 x4 <= 25
bounds
 0 <= x1 <= +infinity
 0 <= x2 <= 10
 0 <= x3 <= +infinity
 0 <= x4 <= +infinity
end
```

### 19.1.7  Mixed integer example `milo1.lp`

```
maximize
obj: x1 + 6.4e-01 x2
subject to
c1:  5e+01 x1 + 3.1e+01 x2 <= 2.5e+02
c2:  3e+00 x1 - 2e+00 x2 >= -4e+00
bounds
 0 <= x1 <= +infinity
 0 <= x2 <= +infinity
general
 x1 x2
end
```

## 19.2  LP format peculiarities

### 19.2.1  Comments

Anything on a line after a "\" is ignored and is treated as a comment.

### 19.2.2  Names

A name for an objective, a constraint or a variable may contain the letters a-z, A-Z, the digits 0-9 and the characters

```
!"#$%&()/,.;?@_'`|~
```

The first character in a name must not be a number, a period or the letter 'e' or 'E'. Keywords must not be used as names.

MOSEK accepts any character as valid for names, except '\0'. When writing a name that is not allowed in LP files, it is changed and a warning is issued.

The algorithm for making names LP valid works as follows: The name is interpreted as an `utf-8` string. For a unicode character `c`:

- If `c=='_'` (underscore), the output is '`__`' (two underscores).

- If `c` is a valid LP name character, the output is just `c`.

- If `c` is another character in the ASCII range, the output is `_XX`, where `XX` is the hexadecimal code for the character.

- If `c` is a character in the range 127—65535, the output is `_uXXXX`, where `XXXX` is the hexadecimal code for the character.

- If `c` is a character above 65535, the output is `_UXXXXXXXX`, where `XXXXXXXX` is the hexadecimal code for the character.

Invalid `utf-8` substrings are escaped as '`_XX`', and if a name starts with a period, 'e' or 'E', that character is escaped as '`_XX`'.

## 19.2.3 Variable bounds

Specifying several upper or lower bounds on one variable is possible but MOSEK uses only the tightest bounds. If a variable is fixed (with `=`), then it is considered the tightest bound.

## 19.2.4 MOSEK specific extensions to the LP format

Some optimization software packages employ a more strict definition of the LP format that the one used by MOSEK. The limitations imposed by the strict LP format are the following:

- Quadratic terms in the constraints are not allowed.

- Names can be only 16 characters long.

- Lines must not exceed 255 characters in length.

If an LP formatted file created by MOSEK should satisfies the strict definition, then the parameter

`MSK_IPAR_WRITE_LP_STRICT_FORMAT`

should be set; note, however, that some problems cannot be written correctly as a strict LP formatted file. For instance, all names are truncated to 16 characters and hence they may loose their uniqueness and change the problem.

To get around some of the inconveniences converting from other problem formats, MOSEK allows lines to contain 1024 characters and names may have any length (shorter than the 1024 characters).

Internally in MOSEK names may contain any (printable) character, many of which cannot be used in LP names. Setting the parameters

MSK_IPAR_READ_LP_QUOTED_NAMES

and

MSK_IPAR_WRITE_LP_QUOTED_NAMES

allows MOSEK to use quoted names. The first parameter tells MOSEK to remove quotes from quoted names e.g, `"x1"`, when reading LP formatted files. The second parameter tells MOSEK to put quotes around any semi-illegal name (names beginning with a number or a period) and fully illegal name (containing illegal characters). As double quote is a legal character in the LP format, quoting semi-illegal names makes them legal in the pure LP format as long as they are still shorter than 16 characters. Fully illegal names are still illegal in a pure LP file.

## 19.3 The strict LP format

The LP format is not a formal standard and different vendors have slightly different interpretations of the LP format. To make MOSEK's definition of the LP format more compatible with the definitions of other vendors, use the parameter setting

MSK_IPAR_WRITE_LP_STRICT_FORMAT = MSK_ON

This setting may lead to truncation of some names and hence to an invalid LP file. The simple solution to this problem is to use the parameter setting

MSK_IPAR_WRITE_GENERIC_NAMES = MSK_ON

which will cause all names to be renamed systematically in the output file.

## 19.4 Formatting of an LP file

A few parameters control the visual formatting of LP files written by MOSEK in order to make it easier to read the files. These parameters are

MSK_IPAR_WRITE_LP_LINE_WIDTH

MSK_IPAR_WRITE_LP_TERMS_PER_LINE

The first parameter sets the maximum number of characters on a single line. The default value is 80 corresponding roughly to the width of a standard text document.

The second parameter sets the maximum number of terms per line; a term means a sign, a coefficient, and a name (for example ”`+ 42 elephants`”). The default value is 0, meaning that there is no maximum.

## 19.4.1   Speeding up file reading

If the input file should be read as fast as possible using the least amount of memory, then it is important to tell MOSEK how many non-zeros, variables and constraints the problem contains. These values can be set using the parameters

MSK_IPAR_READ_CON

MSK_IPAR_READ_VAR

MSK_IPAR_READ_ANZ

MSK_IPAR_READ_QNZ

## 19.4.2   Unnamed constraints

Reading and writing an LP file with MOSEK may change it superficially. If an LP file contains unnamed constraints or objective these are given their generic names when the file is read (however unnamed constraints in MOSEK are written without names).

# Chapter 20

# The OPF format

The Optimization Problem Format (OPF) is an alternative to LP and MPS files for specifying optimization problems. It is row-oriented, inspired by the CPLEX LP format.

Apart from containing objective, constraints, bounds etc. it may contain complete or partial solutions, comments and extra information relevant for solving the problem. It is designed to be easily read and modified by hand and to be forward compatible with possible future extensions.

## 20.1 Intended use

The OPF file format is meant to replace several other files:

- The LP file format. Any problem that can be written as an LP file can be written as an OPF file to; furthermore it naturally accommodates ranged constraints and variables as well as arbitrary characters in names, fixed expressions in the objective, empty constraints, and conic constraints.

- Parameter files. It is possible to specify integer, double and string parameters along with the problem (or in a separate OPF file).

- Solution files. It is possible to store a full or a partial solution in an OPF file and later reload it.

## 20.2 The file format

The format uses tags to structure data. A simple example with the basic sections may look like this:

```
[comment]
  This is a comment. You may write almost anything here...
[/comment]

# This is a single-line comment.

[objective min 'myobj']
```

```
  x + 3 y + x^2 + 3 y^2 + z + 1
[/objective]

[constraints]
  [con 'con01'] 4 <= x + y  [/con]
[/constraints]

[bounds]
  [b] -10 <= x,y <= 10  [/b]

  [cone quad] x,y,z [/cone]
[/bounds]
```

A scope is opened by a tag of the form `[tag]` and closed by a tag of the form `[/tag]`. An opening tag may accept a list of unnamed and named arguments, for examples

```
[tag value] tag with one unnamed argument [/tag]
[tag arg=value] tag with one named argument in quotes [/tag]
```

Unnamed arguments are identified by their order, while named arguments may appear in any order, but never before an unnamed argument. The `value` can be a quoted, single-quoted or double-quoted text string, i.e.

```
[tag 'value']    single-quoted value [/tag]
[tag arg='value'] single-quoted value [/tag]
[tag "value"]    double-quoted value [/tag]
[tag arg="value"] double-quoted value [/tag]
```

### 20.2.1   Sections

The recognized tags are

- `[comment]` A comment section. This can contain *almost* any text: Between single quotes (`'`) or double quotes (`"`) any text may appear. Outside quotes the markup characters (`[` and `]`) must be prefixed by backslashes. Both single and double quotes may appear alone or inside a pair of quotes if it is prefixed by a backslash.

- `[objective]` The objective function: This accepts one or two parameters, where the first one (in the above example 'min') is either `min` or `max` (regardless of case) and defines the objective sense, and the second one (above 'myobj'), if present, is the objective name. The section may contain linear and quadratic expressions.

  If several objectives are specified, all but the last are ignored.

- `[constraints]` This does not directly contain any data, but may contain the subsection 'con' defining a linear constraint.

  `[con]` defines a single constraint; if an argument is present (`[con NAME]`) this is used as the name of the constraint, otherwise it is given a null-name. The section contains a constraint definition written as linear and quadratic expressions with a lower bound, an upper bound, with both or with an equality. Examples:
  ```
  [constraints]
    [con 'con1'] 0 <= x + y      [/con]
    [con 'con2'] 0 >= x + y      [/con]
  ```

```
   [con 'con3'] 0 <= x + y <= 10 [/con]
   [con 'con4']      x + y  = 10 [/con]
 [/constraints]
```

Constraint names are unique. If a constraint is specified which has the same name as a previously defined constraint, the new constraint replaces the existing one.

- [bounds] This does not directly contain any data, but may contain the subsections 'b' (linear bounds on variables) and cone' (quadratic cone).

  - [b]. Bound definition on one or several variables separated by comma (','). An upper or lower bound on a variable replaces any earlier defined bound on that variable. If only one bound (upper or lower) is given only this bound is replaced. This means that upper and lower bounds can be specified separately. So the OPF bound definition:
    ```
    [b]  x,y >= -10  [/b]
    [b]  x,y <= 10   [/b]
    ```
    results in the bound

    $$-10 \le x, y \le 10.$$

  - [cone]. Currently, the supported cones are the *quadratic cone* and the *rotated quadratic cone* A conic constraint is defined as a set of variables which belongs to a single unique cone. A quadratic cone of $n$ variables $x_1, \ldots, x_n$ defines a constraint of the form

    $$x_1^2 > \sum_{i=2}^{n} x_i^2.$$

    A rotated quadratic cone of $n$ variables $x_1, \ldots, x_n$ defines a constraint of the form

    $$x_1 x_2 > \sum_{i=3}^{n} x_i^2.$$

A [bounds]-section example:

```
[bounds]
  [b]  0 <= x,y <= 10  [/b] # ranged bound
  [b] 10 >= x,y >=  0  [/b] # ranged bound
  [b]  0 <= x,y <= inf [/b] # using inf
  [b]       x,y free   [/b] # free variables
  # Let (x,y,z,w) belong to the cone K
  [cone quad]  x,y,z,w  [/cone] # quadratic cone
  [cone rquad] x,y,z,w  [/cone] # rotated quadratic cone
[/bounds]
```

By default all variables are free.

- [variables] This defines an ordering of variables as they should appear in the problem. This is simply a space-separated list of variable names.

- [integer] This contains a space-separated list of variables and defines the constraint that the listed variables must be integer values.

- [hints] This may contain only non-essential data; for example estimates of the number of variables, constraints and non-zeros. Placed before all other sections containing data this may reduce the time spent reading the file.

  In the hints section, any subsection which is not recognized by MOSEK is simply ignored. In this section a hint in a subsection is defined as follows:

  ```
  [hint ITEM] value [/hint]
  ```

  where ITEM may be replaced by numvar (number of variables), numcon (number of linear/quadratic constraints), numanz (number of linear non-zeros in constraints) and numqnz (number of quadratic non-zeros in constraints).

- [solutions] This section can contain a set of full or partial solutions to a problem. Each solution must be specified using a [solution]-section, i.e.

  ```
  [solutions]
          [solution]...[/solution] #solution 1
          [solution]...[/solution] #solution 2
                            #other solutions....
          [solution]...[/solution] #solution n
  [/solutions]
  ```

  Note that a [solution]-section must be always specified inside a [solutions]-section. The syntax of a [solution]-section is the following:

  ```
  [solution SOLTYPE status=STATUS]...[/solution]
  ```

  where SOLTYPE is one of the strings

  - 'interior', a non-basic solution,
  - 'basic', a basic solution,
  - 'integer', an integer solution,

  and STATUS is one of the strings

  - 'UNKNOWN',
  - 'OPTIMAL',
  - 'INTEGER_OPTIMAL',
  - 'PRIM_FEAS',
  - 'DUAL_FEAS',
  - 'PRIM_AND_DUAL_FEAS',
  - 'NEAR_OPTIMAL',
  - 'NEAR_PRIM_FEAS',
  - 'NEAR_DUAL_FEAS',
  - 'NEAR_PRIM_AND_DUAL_FEAS',
  - 'PRIM_INFEAS_CER',
  - 'DUAL_INFEAS_CER',
  - 'NEAR_PRIM_INFEAS_CER',

- 'NEAR_DUAL_INFEAS_CER',
- 'NEAR_INTEGER_OPTIMAL'.

Most of these values are irrelevant for input solutions; when constructing a solution for simplex hot-start or an initial solution for a mixed integer problem the safe setting is UNKNOWN.

A [solution]-section contains [con] and [var] sections. Each [con] and [var] section defines solution information for a single variable or constraint, specified as list of KEYWORD/value pairs, in any order, written as

KEYWORD=value

Allowed keywords are as follows:

- sk. The status of the item, where the value is one of the following strings:
  * LOW, the item is on its lower bound.
  * UPR, the item is on its upper bound.
  * FIX, it is a fixed item.
  * BAS, the item is in the basis.
  * SUPBAS, the item is super basic.
  * UNK, the status is unknown.
  * INF, the item is outside its bounds (infeasible).
- lvl Defines the level of the item.
- sl Defines the level of the dual variable associated with its lower bound.
- su Defines the level of the dual variable associated with its upper bound.
- sn Defines the level of the variable associated with its cone.
- y Defines the level of the corresponding dual variable (for constraints only).

A [var] section should always contain the items sk, lvl, sl and su. Items sl and su are not required for integer solutions.

A [con] section should always contain sk, lvl, sl, su and y.

An example of a solution section

```
[solution basic status=UNKNOWN]
    [var x0] sk=LOW    lvl=5.0       [/var]
    [var x1] sk=UPR    lvl=10.0      [/var]
    [var x2] sk=SUPBAS lvl=2.0  sl=1.5 su=0.0 [/var]

    [con c0] sk=LOW    lvl=3.0 y=0.0 [/con]
    [con c0] sk=UPR    lvl=0.0 y=5.0 [/con]
[/solution]
```

- [vendor] This contains solver/vendor specific data. It accepts one argument, which is a vendor ID – for MOSEK the ID is simply mosek – and the section contains the subsection parameters defining solver parameters. When reading a vendor section, any unknown vendor can be safely ignored. This is described later.

Comments using the '#' may appear anywhere in the file. Between the '#' and the following line-break any text may be written, including markup characters.

## 20.2.2   Numbers

Numbers, when used for parameter values or coefficients, are written in the usual way by the `printf` function. That is, they may be prefixed by a sign (`+` or `-`) and may contain an integer part, decimal part and an exponent. The decimal point is always '`.`' (a dot). Some examples are

```
1
1.0
 .0
1.
1e10
1e+10
1e-10
```

Some *invalid* examples are

```
e10   # invalid, must contain either integer or decimal part
.     # invalid
.e10  # invalid
```

More formally, the following standard regular expression describes numbers as used:

```
[+|-]?([0-9]+[.][0-9]*|[.][0-9]+)([eE][+|-]?[0-9]+)?
```

## 20.2.3   Names

Variable names, constraint names and objective name may contain arbitrary characters, which in some cases must be enclosed by quotes (single or double) that in turn must be preceded by a backslash. Unquoted names must begin with a letter (`a-z` or `A-Z`) and contain only the following characters: the letters `a-z` and `A-Z`, the digits `0-9`, braces (`{` and `}`) and underscore (`_`).

Some examples of legal names:

```
an_unquoted_name
another_name{123}
'single quoted name'
"double quoted name"
"name with \\"quote\\" in it"
"name with []s in it"
```

# 20.3   Parameters section

In the `vendor` section solver parameters are defined inside the `parameters` subsection. Each parameter is written as

```
[p PARAMETER_NAME] value [/p]
```

where `PARAMETER_NAME` is replaced by a MOSEK parameter name, usually of the form `MSK_IPAR_...`, `MSK_DPAR_...` or `MSK_SPAR_...`, and the `value` is replaced by the value of that parameter; both integer values and named values may be used. Some simple examples are:

```
[vendor mosek]
  [parameters]
    [p MSK_IPAR_OPF_MAX_TERMS_PER_LINE] 10      [/p]
```

```
    [p MSK_IPAR_OPF_WRITE_PARAMETERS]   MSK_ON [/p]
    [p MSK_DPAR_DATA_TOL_BOUND_INF]     1.0e18 [/p]
  [/parameters]
[/vendor]
```

## 20.4 Writing OPF files from MOSEK

To write an OPF file set the parameter `MSK_IPAR_WRITE_DATA_FORMAT` to `MSK_DATA_FORMAT_OP` as this ensures that `OPF` format is used. Then modify the following parameters to define what the file should contain:

- `MSK_IPAR_OPF_WRITE_HEADER`, include a small header with comments.

- `MSK_IPAR_OPF_WRITE_HINTS`, include hints about the size of the problem.

- `MSK_IPAR_OPF_WRITE_PROBLEM`, include the problem itself — objective, constraints and bounds.

- `MSK_IPAR_OPF_WRITE_SOLUTIONS`, include solutions if they are defined. If this is off, no solutions are included.

- `MSK_IPAR_OPF_WRITE_SOL_BAS`, include basic solution, if defined.

- `MSK_IPAR_OPF_WRITE_SOL_ITG`, include integer solution, if defined.

- `MSK_IPAR_OPF_WRITE_SOL_ITR`, include interior solution, if defined.

- `MSK_IPAR_OPF_WRITE_PARAMETERS`, include all parameter settings.

## 20.5 Examples

This section contains a set of small examples written in OPF and describing how to formulate linear, quadratic and conic problems.

### 20.5.1 Linear example `lo1.opf`

Consider the example:

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

having the bounds

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

In the OPF format the example is displayed as shown below:

```
[comment]
  The lo1 example in OPF format
[/comment]

[hints]
  [hint NUMVAR] 4 [/hint]
  [hint NUMCON] 3 [/hint]
  [hint NUMANZ] 9 [/hint]
[/hints]

[variables disallow_new_variables]
  x1 x2 x3 x4
[/variables]

[objective maximize 'obj']
   3 x1 + x2 + 5 x3 + x4
[/objective]

[constraints]
  [con 'c1'] 3 x1 +   x2 + 2 x3          = 30 [/con]
  [con 'c2'] 2 x1 +   x2 + 3 x3 +   x4 >= 15 [/con]
  [con 'c3']         2 x2        + 3 x4 <= 25 [/con]
[/constraints]

[bounds]
  [b] 0 <= * [/b]
  [b] 0 <= x2 <= 10 [/b]
[/bounds]
```

## 20.5.2   Quadratic example `qo1.opf`

An example of a quadratic optimization problem is

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

This can be formulated in `opf` as shown below.

```
[comment]
  The qo1 example in OPF format
[/comment]

[hints]
  [hint NUMVAR] 3 [/hint]
  [hint NUMCON] 1 [/hint]
  [hint NUMANZ] 3 [/hint]
  [hint NUMQNZ] 4 [/hint]
```

```
[/hints]

[variables disallow_new_variables]
  x1 x2 x3
[/variables]

[objective minimize 'obj']
  # The quadratic terms are often written with a factor of 1/2 as here,
  # but this is not required.

  - x2 + 0.5 ( 2.0 x1 ^ 2 - 2.0 x3 * x1 + 0.2 x2 ^ 2 + 2.0 x3 ^ 2 )
[/objective]

[constraints]
  [con 'c1'] 1.0 <= x1 + x2 + x3 [/con]
[/constraints]

[bounds]
  [b] 0 <= * [/b]
[/bounds]
```

### 20.5.3   Conic quadratic example `cqo1.opf`

Consider the example:

$$
\begin{array}{lrcl}
\text{minimize} & x_3 + x_4 + x_5 \\
\text{subject to} & x_0 + x_1 + 2x_2 & = & 1, \\
& x_0, x_1, x_2 & \geq & 0, \\
& x_3 \geq \sqrt{x_0^2 + x_1^2}, \\
& 2x_4 x_5 \geq x_2^2.
\end{array}
$$

Please note that the type of the cones is defined by the parameter to [`cone ...`]; the content of the `cone`-section is the names of variables that belong to the cone.

```
[comment]
  The cqo1 example in OPF format.
[/comment]

[hints]
  [hint NUMVAR] 6 [/hint]
  [hint NUMCON] 1 [/hint]
  [hint NUMANZ] 3 [/hint]
[/hints]

[variables disallow_new_variables]
  x1 x2 x3 x4 x5 x6
[/variables]

[objective minimize 'obj']
  x4 + x5 + x6
[/objective]

[constraints]
  [con 'c1']  x1 + x2 + 2e+00 x3 = 1e+00 [/con]
```

```
[/constraints]

[bounds]
  # We let all variables default to the positive orthant
  [b] 0 <= * [/b]

  # ...and change those that differ from the default
  [b] x4,x5,x6 free [/b]

  # Define quadratic cone: x4 >= sqrt( x1^2 + x2^2 )
  [cone quad 'k1'] x4, x1, x2 [/cone]

  # Define rotated quadratic cone: 2 x5 x6 >= x3^2
  [cone rquad 'k2'] x5, x6, x3 [/cone]
[/bounds]
```

### 20.5.4   Mixed integer example `milo1.opf`

Consider the mixed integer problem:

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

This can be implemented in `OPF` with:

```
[comment]
  The milo1 example in OPF format
[/comment]

[hints]
  [hint NUMVAR] 2 [/hint]
  [hint NUMCON] 2 [/hint]
  [hint NUMANZ] 4 [/hint]
[/hints]

[variables disallow_new_variables]
  x1 x2
[/variables]

[objective maximize 'obj']
   x1 + 6.4e-1 x2
[/objective]

[constraints]
  [con 'c1'] 5e+1 x1 + 3.1e+1 x2 <= 2.5e+2 [/con]
  [con 'c2'] -4 <= 3 x1 - 2 x2 [/con]
[/constraints]

[bounds]
  [b] 0 <= * [/b]
[/bounds]

[integer]
```

```
  x1 x2
[/integer]
```

# Chapter 21

# 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.

# Chapter 22

# The XML (OSiL) format

MOSEK can write data in the standard `OSiL` xml format. For a definition of the `OSiL` format please see http://www.optimizationservices.org/. Only linear constraints (possibly with integer variables) are supported. By default output files with the extension `.xml` are written in the `OSiL` format.

The parameter `MSK_IPAR_WRITE_XML_MODE` controls if the linear coefficients in the $A$ matrix are written in row or column order.

# Chapter 23

# Parameters

Parameters grouped by functionality.

Analysis parameters.

Parameters controling the behaviour of the problem and solution analyzers.

- **MSK␣DPAR␣ANA␣SOL␣INFEAS␣TOL**. If a constraint violates its bound with an amount larger than this value, the constraint name, index and violation will be printed by the solution analyzer.

Basis identification parameters.

- **MSK␣IPAR␣BI␣CLEAN␣OPTIMIZER**. Controls which simplex optimizer is used in the clean-up phase.
- **MSK␣IPAR␣BI␣IGNORE␣MAX␣ITER**. Turns on basis identification in case the interior-point optimizer is terminated due to maximum number of iterations.
- **MSK␣IPAR␣BI␣IGNORE␣NUM␣ERROR**. Turns on basis identification in case the interior-point optimizer is terminated due to a numerical problem.
- **MSK␣IPAR␣BI␣MAX␣ITERATIONS**. Maximum number of iterations after basis identification.
- **MSK␣IPAR␣INTPNT␣BASIS**. Controls whether basis identification is performed.
- **MSK␣IPAR␣LOG␣BI**. Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.
- **MSK␣IPAR␣LOG␣BI␣FREQ**. Controls the logging frequency.
- **MSK␣DPAR␣SIM␣LU␣TOL␣REL␣PIV**. Relative pivot tolerance employed when computing the LU factorization of the basis matrix.

Behavior of the optimization task.

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

- **MSK␣SPAR␣FEASREPAIR␣NAME␣PREFIX**. Feasibility repair name prefix.

243

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

Conic interior-point method parameters.

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

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

Data check parameters.

These parameters defines data checking settings and problem data tolerances, i.e. which values are rounded to 0 or infinity, and which values are large or small enough to produce a warning.

- MSK_DPAR_DATA_TOL_AIJ. Data tolerance threshold.
- MSK_DPAR_DATA_TOL_AIJ_HUGE. Data tolerance threshold.
- MSK_DPAR_DATA_TOL_AIJ_LARGE. Data tolerance threshold.
- MSK_DPAR_DATA_TOL_BOUND_INF. Data tolerance threshold.
- MSK_DPAR_DATA_TOL_BOUND_WRN. Data tolerance threshold.
- MSK_DPAR_DATA_TOL_C_HUGE. Data tolerance threshold.
- MSK_DPAR_DATA_TOL_CJ_LARGE. Data tolerance threshold.
- MSK_DPAR_DATA_TOL_QIJ. Data tolerance threshold.
- MSK_DPAR_DATA_TOL_X. Data tolerance threshold.
- MSK_IPAR_LOG_CHECK_CONVEXITY. Controls logging in convexity check on quadratic problems. Set to a positive value to turn logging on.

  If a quadratic coefficient matrix is found to violate the requirement of PSD (NSD) then a list of negative (positive) pivot elements is printed. The absolute value of the pivot elements is also shown.

Data input/output parameters.

Parameters defining the behavior of data readers and writers.

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

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

Debugging parameters.

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

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

Dual simplex optimizer parameters.

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

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

Feasibility repair parameters.

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

Infeasibility report parameters.

- MSK_IPAR_LOG_INFEAS_ANA. Controls log level for the infeasibility analyzer.

Interior-point method parameters.

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

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

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

License manager parameters.

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

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

Logging parameters.

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

Mixed-integer optimization parameters.

- `MSK_IPAR_LOG_MIO`. Controls the amount of log information from the mixed-integer optimizers.

- `MSK_IPAR_LOG_MIO_FREQ`. The mixed-integer solver logging frequency.

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

- `MSK_IPAR_MIO_CONSTRUCT_SOL`. Controls if an initial mixed integer solution should be constructed from the values of the integer variables.

- `MSK_IPAR_MIO_CONT_SOL`. Controls the meaning of interior-point and basic solutions in mixed integer problems.

- `MSK_IPAR_MIO_CUT_CG`. Controls whether CG cuts should be generated.

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

- `MSK_IPAR_MIO_CUT_LEVEL_ROOT`. Controls the cut level employed by the mixed-integer optimizer at the root node.

- `MSK_IPAR_MIO_CUT_LEVEL_TREE`. Controls the cut level employed by the mixed-integer optimizer in the tree.

- `MSK_DPAR_MIO_DISABLE_TERM_TIME`. Certain termination criteria is disabled within the mixed-integer optimizer for period time specified by the parameter.

- `MSK_IPAR_MIO_FEASPUMP_LEVEL`. Controls the feasibility pump heuristic which is used to construct a good initial feasible solution.

- `MSK_IPAR_MIO_HEURISTIC_LEVEL`. Controls the heuristic employed by the mixed-integer optimizer to locate an initial integer feasible solution.

- `MSK_DPAR_MIO_HEURISTIC_TIME`. Time limit for the mixed-integer heuristics.

- `MSK_IPAR_MIO_HOTSTART`. Controls whether the integer optimizer is hot-started.

- `MSK_IPAR_MIO_KEEP_BASIS`. Controls whether the integer presolve keeps bases in memory.

- `MSK_IPAR_MIO_MAX_NUM_BRANCHES`. Maximum number of branches allowed during the branch and bound search.

- `MSK_IPAR_MIO_MAX_NUM_RELAXS`. Maximum number of relaxations in branch and bound search.

- `MSK_IPAR_MIO_MAX_NUM_SOLUTIONS`. Controls how many feasible solutions the mixed-integer optimizer investigates.

- `MSK_DPAR_MIO_MAX_TIME`. Time limit for the mixed-integer optimizer.

- `MSK_DPAR_MIO_MAX_TIME_APRX_OPT`. Time limit for the mixed-integer optimizer.

- `MSK_DPAR_MIO_NEAR_TOL_ABS_GAP`. Relaxed absolute optimality tolerance employed by the mixed-integer optimizer.

- `MSK_DPAR_MIO_NEAR_TOL_REL_GAP`. The mixed-integer optimizer is terminated when this tolerance is satisfied.

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

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

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

Network simplex optimizer parameters.

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

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

Non-convex solver parameters.

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

Nonlinear convex method parameters.

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

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

  If a quadratic coefficient matrix is found to violate the requirement of PSD (NSD) then a list of negative (positive) pivot elements is printed. The absolute value of the pivot elements is also shown.

Optimization system parameters.

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

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

Output information parameters.

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

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

Overall solver parameters.

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

- `MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX`. Priority of the free simplex optimizer when selecting solvers for concurrent optimization.

- `MSK_IPAR_CONCURRENT_PRIORITY_INTPNT`. Priority of the interior-point algorithm when selecting solvers for concurrent optimization.

- `MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX`. Priority of the primal simplex algorithm when selecting solvers for concurrent optimization.

- `MSK_IPAR_INFEAS_PREFER_PRIMAL`. Controls which certificate is used if both primal- and dual- certificate of infeasibility is available.

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

- `MSK_IPAR_MIO_CONT_SOL`. Controls the meaning of interior-point and basic solutions in mixed integer problems.

- `MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER`. Controls the size of the local search space when doing local branching.

- `MSK_IPAR_MIO_MODE`. Turns on/off the mixed-integer mode.

- `MSK_IPAR_OPTIMIZER`. Controls which optimizer is used to optimize the task.

- `MSK_IPAR_PRESOLVE_LEVEL`. Currently not used.

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

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

Presolve parameters.

- `MSK_IPAR_PRESOLVE_ELIM_FILL`. Maximum amount of fill-in in the elimination phase.

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

- `MSK_IPAR_PRESOLVE_ELIMINATOR_USE`. Controls whether free or implied free variables are eliminated from the problem.

- `MSK_IPAR_PRESOLVE_LEVEL`. Currently not used.

- `MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH`. Controls linear dependency check in presolve.

- `MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH`. Controls linear dependency check in presolve.

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

- `MSK_DPAR_PRESOLVE_TOL_ABS_LINDEP`. Absolute tolerance employed by the linear dependency checker.

- `MSK_DPAR_PRESOLVE_TOL_AIJ`. Absolute zero tolerance employed for constraint coefficients in the presolve.

- `MSK_DPAR_PRESOLVE_TOL_REL_LINDEP`. Relative tolerance employed by the linear dependency checker.

- `MSK_DPAR_PRESOLVE_TOL_S`. Absolute zero tolerance employed for slack variables in the presolve.

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

Primal simplex optimizer parameters.

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

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

Progress call-back parameters.

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

Simplex optimizer parameters.

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

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

- `MSK_IPAR_SIM_MAX_NUM_SETBACKS`. Controls how many set-backs that are allowed within a simplex optimizer.

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

- `MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD`. An exprimental feature.

- `MSK_IPAR_SIM_REFORMULATION`. Controls if the simplex optimizers are allowed to reformulate the problem.

- `MSK_IPAR_SIM_SAVE_LU`. Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.

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

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

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

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

- `MSK_IPAR_SIM_SWITCH_OPTIMIZER`. Controls the simplex behavior.

- `MSK_DPAR_SIMPLEX_ABS_TOL_PIV`. Absolute pivot tolerance employed by the simplex optimizers.

Solution input/output parameters.

Parameters defining the behavior of solution reader and writer.

- `MSK_SPAR_BAS_SOL_FILE_NAME`. Name of the bas solution file.

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

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

- `MSK_IPAR_SOL_FILTER_KEEP_BASIC`. Controls the license manager client behavior.

- `MSK_SPAR_SOL_FILTER_XC_LOW`. Solution file filter.

- `MSK_SPAR_SOL_FILTER_XC_UPR`. Solution file filter.

- `MSK_SPAR_SOL_FILTER_XX_LOW`. Solution file filter.

- `MSK_SPAR_SOL_FILTER_XX_UPR`. Solution file filter.

Termination criterion parameters.

Parameters which define termination and optimality criteria and related information.

- `MSK_DPAR_BASIS_REL_TOL_S`. Maximum relative dual bound violation allowed in an optimal basic solution.

- `MSK_DPAR_BASIS_TOL_S`. Maximum absolute dual bound violation in an optimal basic solution.

- `MSK_DPAR_BASIS_TOL_X`. Maximum absolute primal bound violation allowed in an optimal basic solution.

- `MSK_IPAR_BI_MAX_ITERATIONS`. Maximum number of iterations after basis identification.

- `MSK_DPAR_INTPNT_CO_TOL_DFEAS`. Dual feasibility tolerance used by the conic interior-point optimizer.

- `MSK_DPAR_INTPNT_CO_TOL_INFEAS`. Infeasibility tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_MU_RED`. Optimality tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_NEAR_REL`. Optimality tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_PFEAS`. Primal feasibility tolerance used by the conic interior-point optimizer.

- `MSK_DPAR_INTPNT_CO_TOL_REL_GAP`. Relative gap termination tolerance used by the conic interior-point optimizer.

- `MSK_IPAR_INTPNT_MAX_ITERATIONS`. Controls the maximum number of iterations allowed in the interior-point optimizer.

- `MSK_DPAR_INTPNT_NL_TOL_DFEAS`. Dual feasibility tolerance used when a nonlinear model is solved.

- `MSK_DPAR_INTPNT_NL_TOL_MU_RED`. Relative complementarity gap tolerance.

- `MSK_DPAR_INTPNT_NL_TOL_NEAR_REL`. Nonlinear solver optimality tolerance parameter.

- `MSK_DPAR_INTPNT_NL_TOL_PFEAS`. Primal feasibility tolerance used when a nonlinear model is solved.

- `MSK_DPAR_INTPNT_NL_TOL_REL_GAP`. Relative gap termination tolerance for nonlinear problems.

- `MSK_DPAR_INTPNT_TOL_DFEAS`. Dual feasibility tolerance used for linear and quadratic optimization problems.

- `MSK_DPAR_INTPNT_TOL_INFEAS`. Nonlinear solver infeasibility tolerance parameter.

- `MSK_DPAR_INTPNT_TOL_MU_RED`. Relative complementarity gap tolerance.

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

- `MSK_DPAR_INTPNT_TOL_REL_GAP`. Relative gap termination tolerance.

- `MSK_DPAR_LOWER_OBJ_CUT`. Objective bound.

- `MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH`. Objective bound.

- `MSK_DPAR_MIO_DISABLE_TERM_TIME`. Certain termination criteria is disabled within the mixed-integer optimizer for period time specified by the parameter.

- `MSK_IPAR_MIO_MAX_NUM_BRANCHES`. Maximum number of branches allowed during the branch and bound search.

- `MSK_IPAR_MIO_MAX_NUM_SOLUTIONS`. Controls how many feasible solutions the mixed-integer optimizer investigates.

- `MSK_DPAR_MIO_MAX_TIME`. Time limit for the mixed-integer optimizer.

- `MSK_DPAR_MIO_NEAR_TOL_REL_GAP`. The mixed-integer optimizer is terminated when this tolerance is satisfied.

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

- Integer parameters
- Double parameters
- String parameters

## 23.1  MSKdparame: Double parameters

### 23.1.1  MSK_DPAR_ANA_SOL_INFEAS_TOL

**Corresponding constant:**

MSK_DPAR_ANA_SOL_INFEAS_TOL

**Description:**

If a constraint violates its bound with an amount larger than this value, the constraint name, index and violation will be printed by the solution analyzer.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1e-6

### 23.1.2  MSK_DPAR_BASIS_REL_TOL_S

**Corresponding constant:**

MSK_DPAR_BASIS_REL_TOL_S

**Description:**

Maximum relative dual bound violation allowed in an optimal basic solution.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

    1.0e-12

### 23.1.3   MSK_DPAR_BASIS_TOL_S

**Corresponding constant:**

    MSK_DPAR_BASIS_TOL_S

**Description:**

Maximum absolute dual bound violation in an optimal basic solution.

**Possible Values:**

Any number between 1.0e-9 and +inf.

**Default value:**

    1.0e-6

### 23.1.4   MSK_DPAR_BASIS_TOL_X

**Corresponding constant:**

    MSK_DPAR_BASIS_TOL_X

**Description:**

Maximum absolute primal bound violation allowed in an optimal basic solution.

**Possible Values:**

Any number between 1.0e-9 and +inf.

**Default value:**

    1.0e-6

### 23.1.5   MSK_DPAR_CHECK_CONVEXITY_REL_TOL

**Corresponding constant:**

    MSK_DPAR_CHECK_CONVEXITY_REL_TOL

**Description:**

This parameter controls when the full convexity check declares a problem to be non-convex. Increasing this tolerance relaxes the criteria for declaring the problem non-convex.

A problem is declared non-convex if negative (positive) pivot elements are detected in the cholesky factor of a matrix which is required to be PSD (NSD). This parameter controles how much this non-negativity requirement may be violated.

If $d_i$ is the pivot element for column $i$, then the matrix $Q$ is considered to not be PSD if:

$$d_i \leq - |Q_{ii}| * \texttt{check\_convexity\_rel\_tol}$$

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1e-10

## 23.1.6  MSK_DPAR_DATA_TOL_AIJ

**Corresponding constant:**

MSK_DPAR_DATA_TOL_AIJ

**Description:**

Absolute zero tolerance for elements in $A$. If any value $A_{ij}$ is smaller than this parameter in absolute terms MOSEK will treat the values as zero and generate a warning.

**Possible Values:**

Any number between 1.0e-16 and 1.0e-6.

**Default value:**

1.0e-12

## 23.1.7  MSK_DPAR_DATA_TOL_AIJ_HUGE

**Corresponding constant:**

MSK_DPAR_DATA_TOL_AIJ_HUGE

**Description:**

An element in $A$ which is larger than this value in absolute size causes an error.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e20

## 23.1.8   MSK_DPAR_DATA_TOL_AIJ_LARGE

**Corresponding constant:**

MSK_DPAR_DATA_TOL_AIJ_LARGE

**Description:**

An element in $A$ which is larger than this value in absolute size causes a warning message to be printed.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e10

## 23.1.9   MSK_DPAR_DATA_TOL_BOUND_INF

**Corresponding constant:**

MSK_DPAR_DATA_TOL_BOUND_INF

**Description:**

Any bound which in absolute value is greater than this parameter is considered infinite.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e16

## 23.1.10   MSK_DPAR_DATA_TOL_BOUND_WRN

**Corresponding constant:**

MSK_DPAR_DATA_TOL_BOUND_WRN

**Description:**

If a bound value is larger than this value in absolute size, then a warning message is issued.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e8

## 23.1.11   MSK_DPAR_DATA_TOL_C_HUGE

**Corresponding constant:**

MSK_DPAR_DATA_TOL_C_HUGE

**Description:**

An element in $c$ which is larger than the value of this parameter in absolute terms is considered to be huge and generates an error.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e16

## 23.1.12   MSK_DPAR_DATA_TOL_CJ_LARGE

**Corresponding constant:**

MSK_DPAR_DATA_TOL_CJ_LARGE

**Description:**

An element in $c$ which is larger than this value in absolute terms causes a warning message to be printed.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e8

## 23.1.13   MSK_DPAR_DATA_TOL_QIJ

**Corresponding constant:**

MSK_DPAR_DATA_TOL_QIJ

**Description:**

Absolute zero tolerance for elements in $Q$ matrixes.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e-16

### 23.1.14   MSK_DPAR_DATA_TOL_X

**Corresponding constant:**

MSK_DPAR_DATA_TOL_X

**Description:**

Zero tolerance for constraints and variables i.e. if the distance between the lower and upper bound is less than this value, then the lower and lower bound is considered identical.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e-8

### 23.1.15   MSK_DPAR_FEASREPAIR_TOL

**Corresponding constant:**

MSK_DPAR_FEASREPAIR_TOL

**Description:**

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

**Possible Values:**

Any number between 1.0e-16 and 1.0e+16.

**Default value:**

1.0e-10

### 23.1.16   MSK_DPAR_INTPNT_CO_TOL_DFEAS

**Corresponding constant:**

MSK_DPAR_INTPNT_CO_TOL_DFEAS

**Description:**

Dual feasibility tolerance used by the conic interior-point optimizer.

**Possible Values:**

Any number between 0.0 and 1.0.

**Default value:**

1.0e-8

**See also:**

- MSK_DPAR_INTPNT_CO_TOL_NEAR_REL Optimality tolerance for the conic solver.

### 23.1.17 MSK_DPAR_INTPNT_CO_TOL_INFEAS

**Corresponding constant:**

MSK_DPAR_INTPNT_CO_TOL_INFEAS

**Description:**

Controls when the conic interior-point optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

**Possible Values:**

Any number between 0.0 and 1.0.

**Default value:**

1.0e-10

### 23.1.18 MSK_DPAR_INTPNT_CO_TOL_MU_RED

**Corresponding constant:**

MSK_DPAR_INTPNT_CO_TOL_MU_RED

**Description:**

Relative complementarity gap tolerance feasibility tolerance used by the conic interior-point optimizer.

**Possible Values:**

Any number between 0.0 and 1.0.

**Default value:**

1.0e-8

### 23.1.19 MSK_DPAR_INTPNT_CO_TOL_NEAR_REL

**Corresponding constant:**

MSK_DPAR_INTPNT_CO_TOL_NEAR_REL

**Description:**

If MOSEK cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.

**Possible Values:**

Any number between 1.0 and +inf.

**Default value:**

1000

### 23.1.20    MSK_DPAR_INTPNT_CO_TOL_PFEAS

**Corresponding constant:**

MSK_DPAR_INTPNT_CO_TOL_PFEAS

**Description:**

Primal feasibility tolerance used by the conic interior-point optimizer.

**Possible Values:**

Any number between 0.0 and 1.0.

**Default value:**

1.0e-8

**See also:**

- MSK_DPAR_INTPNT_CO_TOL_NEAR_REL Optimality tolerance for the conic solver.

### 23.1.21    MSK_DPAR_INTPNT_CO_TOL_REL_GAP

**Corresponding constant:**

MSK_DPAR_INTPNT_CO_TOL_REL_GAP

**Description:**

Relative gap termination tolerance used by the conic interior-point optimizer.

**Possible Values:**

Any number between 0.0 and 1.0.

**Default value:**

1.0e-7

**See also:**

- MSK_DPAR_INTPNT_CO_TOL_NEAR_REL Optimality tolerance for the conic solver.

### 23.1.22    MSK_DPAR_INTPNT_NL_MERIT_BAL

**Corresponding constant:**

MSK_DPAR_INTPNT_NL_MERIT_BAL

**Description:**

Controls if the complementarity and infeasibility is converging to zero at about equal rates.

**Possible Values:**

Any number between 0.0 and 0.99.

**Default value:**

    1.0e-4

### 23.1.23 MSK_DPAR_INTPNT_NL_TOL_DFEAS

**Corresponding constant:**

    MSK_DPAR_INTPNT_NL_TOL_DFEAS

**Description:**

Dual feasibility tolerance used when a nonlinear model is solved.

**Possible Values:**

Any number between 0.0 and 1.0.

**Default value:**

    1.0e-8

### 23.1.24 MSK_DPAR_INTPNT_NL_TOL_MU_RED

**Corresponding constant:**

    MSK_DPAR_INTPNT_NL_TOL_MU_RED

**Description:**

Relative complementarity gap tolerance.

**Possible Values:**

Any number between 0.0 and 1.0.

**Default value:**

    1.0e-12

### 23.1.25 MSK_DPAR_INTPNT_NL_TOL_NEAR_REL

**Corresponding constant:**

    MSK_DPAR_INTPNT_NL_TOL_NEAR_REL

**Description:**

If the MOSEK nonlinear interior-point optimizer cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.

**Possible Values:**

Any number between 1.0 and +inf.

**Default value:**

    `1000.0`

### 23.1.26   MSK_DPAR_INTPNT_NL_TOL_PFEAS

**Corresponding constant:**

    MSK_DPAR_INTPNT_NL_TOL_PFEAS

**Description:**

    Primal feasibility tolerance used when a nonlinear model is solved.

**Possible Values:**

    Any number between 0.0 and 1.0.

**Default value:**

    `1.0e-8`

### 23.1.27   MSK_DPAR_INTPNT_NL_TOL_REL_GAP

**Corresponding constant:**

    MSK_DPAR_INTPNT_NL_TOL_REL_GAP

**Description:**

    Relative gap termination tolerance for nonlinear problems.

**Possible Values:**

    Any number between 1.0e-14 and +inf.

**Default value:**

    `1.0e-6`

### 23.1.28   MSK_DPAR_INTPNT_NL_TOL_REL_STEP

**Corresponding constant:**

    MSK_DPAR_INTPNT_NL_TOL_REL_STEP

**Description:**

    Relative step size to the boundary for general nonlinear optimization problems.

**Possible Values:**

    Any number between 1.0e-4 and 0.9999999.

**Default value:**

    `0.995`

### 23.1.29   MSK_DPAR_INTPNT_TOL_DFEAS

**Corresponding constant:**

MSK_DPAR_INTPNT_TOL_DFEAS

**Description:**

Dual feasibility tolerance used for linear and quadratic optimization problems.

**Possible Values:**

Any number between 0.0 and 1.0.

**Default value:**

1.0e-8

### 23.1.30   MSK_DPAR_INTPNT_TOL_DSAFE

**Corresponding constant:**

MSK_DPAR_INTPNT_TOL_DSAFE

**Description:**

Controls the initial dual starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly.

**Possible Values:**

Any number between 1.0e-4 and +inf.

**Default value:**

1.0

### 23.1.31   MSK_DPAR_INTPNT_TOL_INFEAS

**Corresponding constant:**

MSK_DPAR_INTPNT_TOL_INFEAS

**Description:**

Controls when the optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

**Possible Values:**

Any number between 0.0 and 1.0.

**Default value:**

1.0e-10

### 23.1.32   MSK_DPAR_INTPNT_TOL_MU_RED

**Corresponding constant:**

MSK_DPAR_INTPNT_TOL_MU_RED

**Description:**

Relative complementarity gap tolerance.

**Possible Values:**

Any number between 0.0 and 1.0.

**Default value:**

1.0e-16

### 23.1.33   MSK_DPAR_INTPNT_TOL_PATH

**Corresponding constant:**

MSK_DPAR_INTPNT_TOL_PATH

**Description:**

Controls how close the interior-point optimizer follows the central path. A large value of this parameter means the central is followed very closely. On numerical unstable problems it may be worthwhile to increase this parameter.

**Possible Values:**

Any number between 0.0 and 0.9999.

**Default value:**

1.0e-8

### 23.1.34   MSK_DPAR_INTPNT_TOL_PFEAS

**Corresponding constant:**

MSK_DPAR_INTPNT_TOL_PFEAS

**Description:**

Primal feasibility tolerance used for linear and quadratic optimization problems.

**Possible Values:**

Any number between 0.0 and 1.0.

**Default value:**

1.0e-8

### 23.1.35 MSK_DPAR_INTPNT_TOL_PSAFE

**Corresponding constant:**

MSK_DPAR_INTPNT_TOL_PSAFE

**Description:**

Controls the initial primal starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly and/or the constraint or variable bounds are very large, then it may be worthwhile to increase this value.

**Possible Values:**

Any number between 1.0e-4 and +inf.

**Default value:**

`1.0`

### 23.1.36 MSK_DPAR_INTPNT_TOL_REL_GAP

**Corresponding constant:**

MSK_DPAR_INTPNT_TOL_REL_GAP

**Description:**

Relative gap termination tolerance.

**Possible Values:**

Any number between 1.0e-14 and +inf.

**Default value:**

`1.0e-8`

### 23.1.37 MSK_DPAR_INTPNT_TOL_REL_STEP

**Corresponding constant:**

MSK_DPAR_INTPNT_TOL_REL_STEP

**Description:**

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

**Possible Values:**

Any number between 1.0e-4 and 0.999999.

**Default value:**

`0.9999`

### 23.1.38   MSK_DPAR_INTPNT_TOL_STEP_SIZE

**Corresponding constant:**

MSK_DPAR_INTPNT_TOL_STEP_SIZE

**Description:**

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

**Possible Values:**

Any number between 0.0 and 1.0.

**Default value:**

1.0e-6

### 23.1.39   MSK_DPAR_LOWER_OBJ_CUT

**Corresponding constant:**

MSK_DPAR_LOWER_OBJ_CUT

**Description:**

If either a primal or dual feasible solution is found proving that the optimal objective value is outside, the interval [MSK_DPAR_LOWER_OBJ_CUT, MSK_DPAR_UPPER_OBJ_CUT], then MOSEK is terminated.

**Possible Values:**

Any number between -inf and +inf.

**Default value:**

-1.0e30

**See also:**

- MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH  Objective bound.

### 23.1.40   MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH

**Corresponding constant:**

MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH

**Description:**

If the lower objective cut is less than the value of this parameter value, then the lower objective cut i.e. MSK_DPAR_LOWER_OBJ_CUT is treated as $-\infty$.

**Possible Values:**

Any number between -inf and +inf.

**Default value:**

-0.5e30

## 23.1.41 MSK_DPAR_MIO_DISABLE_TERM_TIME

**Corresponding constant:**

MSK_DPAR_MIO_DISABLE_TERM_TIME

**Description:**

The termination criteria governed by

- MSK_IPAR_MIO_MAX_NUM_RELAXS
- MSK_IPAR_MIO_MAX_NUM_BRANCHES
- MSK_DPAR_MIO_NEAR_TOL_ABS_GAP
- MSK_DPAR_MIO_NEAR_TOL_REL_GAP

is disabled the first $n$ seconds. This parameter specifies the number $n$. A negative value is identical to infinity i.e. the termination criteria are never checked.

**Possible Values:**

Any number between -inf and +inf.

**Default value:**

-1.0

**See also:**

- MSK_IPAR_MIO_MAX_NUM_RELAXS Maximum number of relaxations in branch and bound search.
- MSK_IPAR_MIO_MAX_NUM_BRANCHES Maximum number of branches allowed during the branch and bound search.
- MSK_DPAR_MIO_NEAR_TOL_ABS_GAP Relaxed absolute optimality tolerance employed by the mixed-integer optimizer.
- MSK_DPAR_MIO_NEAR_TOL_REL_GAP The mixed-integer optimizer is terminated when this tolerance is satisfied.

## 23.1.42 MSK_DPAR_MIO_HEURISTIC_TIME

**Corresponding constant:**

MSK_DPAR_MIO_HEURISTIC_TIME

**Description:**

Minimum amount of time to be used in the heuristic search for a good feasible integer solution. A negative values implies that the optimizer decides the amount of time to be spent in the heuristic.

**Possible Values:**

Any number between -inf and +inf.

**Default value:**

-1.0

## 23.1.43   MSK_DPAR_MIO_MAX_TIME

**Corresponding constant:**

MSK_DPAR_MIO_MAX_TIME

**Description:**

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

**Possible Values:**

Any number between -inf and +inf.

**Default value:**

-1.0

## 23.1.44   MSK_DPAR_MIO_MAX_TIME_APRX_OPT

**Corresponding constant:**

MSK_DPAR_MIO_MAX_TIME_APRX_OPT

**Description:**

Number of seconds spent by the mixed-integer optimizer before the MSK_DPAR_MIO_TOL_REL_RELAX_INT is applied.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

60

### 23.1.45    MSK_DPAR_MIO_NEAR_TOL_ABS_GAP

**Corresponding constant:**

MSK_DPAR_MIO_NEAR_TOL_ABS_GAP

**Description:**

Relaxed absolute optimality tolerance employed by the mixed-integer optimizer. This termination criteria is delayed. See MSK_DPAR_MIO_DISABLE_TERM_TIME for details.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

0.0

**See also:**

- MSK_DPAR_MIO_DISABLE_TERM_TIME Certain termination criteria is disabled within the mixed-integer optimizer for period time specified by the parameter.

### 23.1.46    MSK_DPAR_MIO_NEAR_TOL_REL_GAP

**Corresponding constant:**

MSK_DPAR_MIO_NEAR_TOL_REL_GAP

**Description:**

The mixed-integer optimizer is terminated when this tolerance is satisfied. This termination criteria is delayed. See MSK_DPAR_MIO_DISABLE_TERM_TIME for details.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e-3

**See also:**

- MSK_DPAR_MIO_DISABLE_TERM_TIME Certain termination criteria is disabled within the mixed-integer optimizer for period time specified by the parameter.

### 23.1.47   MSK_DPAR_MIO_REL_ADD_CUT_LIMITED

**Corresponding constant:**

MSK_DPAR_MIO_REL_ADD_CUT_LIMITED

**Description:**

Controls how many cuts the mixed-integer optimizer is allowed to add to the problem. Let $\alpha$ be the value of this parameter and $m$ the number constraints, then mixed-integer optimizer is allowed to $\alpha m$ cuts.

**Possible Values:**

Any number between 0.0 and 2.0.

**Default value:**

0.75

### 23.1.48   MSK_DPAR_MIO_REL_GAP_CONST

**Corresponding constant:**

MSK_DPAR_MIO_REL_GAP_CONST

**Description:**

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

**Possible Values:**

Any number between 1.0e-15 and +inf.

**Default value:**

1.0e-10

### 23.1.49   MSK_DPAR_MIO_TOL_ABS_GAP

**Corresponding constant:**

MSK_DPAR_MIO_TOL_ABS_GAP

**Description:**

Absolute optimality tolerance employed by the mixed-integer optimizer.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

0.0

### 23.1.50 MSK_DPAR_MIO_TOL_ABS_RELAX_INT

**Corresponding constant:**

MSK_DPAR_MIO_TOL_ABS_RELAX_INT

**Description:**

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

**Possible Values:**

Any number between 1e-9 and +inf.

**Default value:**

1.0e-5

### 23.1.51 MSK_DPAR_MIO_TOL_FEAS

**Corresponding constant:**

MSK_DPAR_MIO_TOL_FEAS

**Description:**

Feasibility tolerance for mixed integer solver. Any solution with maximum infeasibility below this value will be considered feasible.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e-7

### 23.1.52 MSK_DPAR_MIO_TOL_MAX_CUT_FRAC_RHS

**Corresponding constant:**

MSK_DPAR_MIO_TOL_MAX_CUT_FRAC_RHS

**Description:**

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

**Possible Values:**

Any number between 0.0 and 1.0.

**Default value:**

0.0

### 23.1.53   MSK_DPAR_MIO_TOL_MIN_CUT_FRAC_RHS

**Corresponding constant:**

MSK_DPAR_MIO_TOL_MIN_CUT_FRAC_RHS

**Description:**

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

**Possible Values:**

Any number between 0.0 and 1.0.

**Default value:**

0.0

### 23.1.54   MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT

**Corresponding constant:**

MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT

**Description:**

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

**Possible Values:**

Any number between 0.0 and 1.0.

**Default value:**

0.0

### 23.1.55   MSK_DPAR_MIO_TOL_REL_GAP

**Corresponding constant:**

MSK_DPAR_MIO_TOL_REL_GAP

**Description:**

Relative optimality tolerance employed by the mixed-integer optimizer.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e-4

### 23.1.56 MSK_DPAR_MIO_TOL_REL_RELAX_INT

**Corresponding constant:**

MSK_DPAR_MIO_TOL_REL_RELAX_INT

**Description:**

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

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e-6

### 23.1.57 MSK_DPAR_MIO_TOL_X

**Corresponding constant:**

MSK_DPAR_MIO_TOL_X

**Description:**

Absolute solution tolerance used in mixed-integer optimizer.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e-6

### 23.1.58 MSK_DPAR_NONCONVEX_TOL_FEAS

**Corresponding constant:**

MSK_DPAR_NONCONVEX_TOL_FEAS

**Description:**

Feasibility tolerance used by the nonconvex optimizer.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e-6

### 23.1.59   MSK_DPAR_NONCONVEX_TOL_OPT

**Corresponding constant:**

MSK_DPAR_NONCONVEX_TOL_OPT

**Description:**

Optimality tolerance used by the nonconvex optimizer.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e-7

### 23.1.60   MSK_DPAR_OPTIMIZER_MAX_TIME

**Corresponding constant:**

MSK_DPAR_OPTIMIZER_MAX_TIME

**Description:**

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

**Possible Values:**

Any number between -inf and +inf.

**Default value:**

-1.0

### 23.1.61   MSK_DPAR_PRESOLVE_TOL_ABS_LINDEP

**Corresponding constant:**

MSK_DPAR_PRESOLVE_TOL_ABS_LINDEP

**Description:**

Absolute tolerance employed by the linear dependency checker.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e-6

### 23.1.62   MSK_DPAR_PRESOLVE_TOL_AIJ

**Corresponding constant:**

MSK_DPAR_PRESOLVE_TOL_AIJ

**Description:**

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

**Possible Values:**

Any number between 1.0e-15 and +inf.

**Default value:**

1.0e-12

### 23.1.63   MSK_DPAR_PRESOLVE_TOL_REL_LINDEP

**Corresponding constant:**

MSK_DPAR_PRESOLVE_TOL_REL_LINDEP

**Description:**

Relative tolerance employed by the linear dependency checker.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e-10

### 23.1.64   MSK_DPAR_PRESOLVE_TOL_S

**Corresponding constant:**

MSK_DPAR_PRESOLVE_TOL_S

**Description:**

Absolute zero tolerance employed for $s_i$ in the presolve.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e-8

### 23.1.65   MSK_DPAR_PRESOLVE_TOL_X

**Corresponding constant:**

MSK_DPAR_PRESOLVE_TOL_X

**Description:**

Absolute zero tolerance employed for $x_j$ in the presolve.

**Possible Values:**

Any number between 0.0 and +inf.

**Default value:**

1.0e-8

### 23.1.66   MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL

**Corresponding constant:**

MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL

**Description:**

This parameter determines when columns are dropped in incomplete cholesky factorization doing reformulation of quadratic problems.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1e-15

### 23.1.67   MSK_DPAR_SIM_LU_TOL_REL_PIV

**Corresponding constant:**

MSK_DPAR_SIM_LU_TOL_REL_PIV

**Description:**

Relative pivot tolerance employed when computing the LU factorization of the basis in the simplex optimizers and in the basis identification procedure.

A value closer to 1.0 generally improves numerical stability but typically also implies an increase in the computational work.

**Possible Values:**

Any number between 1.0e-6 and 0.999999.

**Default value:**

0.01

## 23.1.68   MSK_DPAR_SIMPLEX_ABS_TOL_PIV

**Corresponding constant:**

MSK_DPAR_SIMPLEX_ABS_TOL_PIV

**Description:**

Absolute pivot tolerance employed by the simplex optimizers.

**Possible Values:**

Any number between 1.0e-12 and +inf.

**Default value:**

1.0e-7

## 23.1.69   MSK_DPAR_UPPER_OBJ_CUT

**Corresponding constant:**

MSK_DPAR_UPPER_OBJ_CUT

**Description:**

If either a primal or dual feasible solution is found proving that the optimal objective value is outside, [MSK_DPAR_LOWER_OBJ_CUT, MSK_DPAR_UPPER_OBJ_CUT], then MOSEK is terminated.

**Possible Values:**

Any number between -inf and +inf.

**Default value:**

1.0e30

**See also:**

- MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH Objective bound.

## 23.1.70   MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH

**Corresponding constant:**

MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH

**Description:**

If the upper objective cut is greater than the value of this value parameter, then the the upper objective cut MSK_DPAR_UPPER_OBJ_CUT is treated as $\infty$.

**Possible Values:**

Any number between -inf and +inf.

**Default value:**

0.5e30

## 23.2   MSKiparame: Integer parameters

### 23.2.1   MSK_IPAR_ALLOC_ADD_QNZ

**Corresponding constant:**

MSK_IPAR_ALLOC_ADD_QNZ

**Description:**

Additional number of $Q$ non-zeros that are allocated space for when `numanz` exceeds `maxnumqnz` during addition of new $Q$ entries.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

5000

### 23.2.2   MSK_IPAR_ANA_SOL_BASIS

**Corresponding constant:**

MSK_IPAR_ANA_SOL_BASIS

**Description:**

Controls whether the basis matrix is analyzed in solaution analyzer.

**Possible values:**

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

**Default value:**

MSK_ON

### 23.2.3   MSK_IPAR_ANA_SOL_PRINT_VIOLATED

**Corresponding constant:**

MSK_IPAR_ANA_SOL_PRINT_VIOLATED

**Description:**

Controls whether a list of violated constraints is printed.

**Possible values:**

- MSK_OFF Switch the option off.

- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

### 23.2.4 MSK_IPAR_AUTO_SORT_A_BEFORE_OPT

**Corresponding constant:**

MSK_IPAR_AUTO_SORT_A_BEFORE_OPT

**Description:**

Controls whether the elements in each column of $A$ are sorted before an optimization is performed. This is not required but makes the optimization more deterministic.

**Possible values:**

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

**Default value:**

MSK_OFF

### 23.2.5 MSK_IPAR_AUTO_UPDATE_SOL_INFO

**Corresponding constant:**

MSK_IPAR_AUTO_UPDATE_SOL_INFO

**Description:**

Controls whether the solution information items are automatically updated after an optimization is performed.

**Possible values:**

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

**Default value:**

MSK_OFF

## 23.2.6   MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE

**Corresponding constant:**

MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE

**Description:**

If a slack variable is in the basis, then the corresponding column in the basis is a unit vector with -1 in the right position. However, if this parameter is set to `MSK_ON`, -1 is replaced by 1.

**Possible values:**

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

**Default value:**

`MSK_OFF`

## 23.2.7   MSK_IPAR_BI_CLEAN_OPTIMIZER

**Corresponding constant:**

MSK_IPAR_BI_CLEAN_OPTIMIZER

**Description:**

Controls which simplex optimizer is used in the clean-up phase.

**Possible values:**

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

**Default value:**

`MSK_OPTIMIZER_FREE`

## 23.2.8    MSK_IPAR_BI_IGNORE_MAX_ITER

**Corresponding constant:**

MSK_IPAR_BI_IGNORE_MAX_ITER

**Description:**

If the parameter MSK_IPAR_INTPNT_BASIS has the value MSK_BI_NO_ERROR and the interior-point optimizer has terminated due to maximum number of iterations, then basis identification is performed if this parameter has the value MSK_ON.

**Possible values:**

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

**Default value:**

MSK_OFF

## 23.2.9    MSK_IPAR_BI_IGNORE_NUM_ERROR

**Corresponding constant:**

MSK_IPAR_BI_IGNORE_NUM_ERROR

**Description:**

If the parameter MSK_IPAR_INTPNT_BASIS has the value MSK_BI_NO_ERROR and the interior-point optimizer has terminated due to a numerical problem, then basis identification is performed if this parameter has the value MSK_ON.

**Possible values:**

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

**Default value:**

MSK_OFF

## 23.2.10    MSK_IPAR_BI_MAX_ITERATIONS

**Corresponding constant:**

MSK_IPAR_BI_MAX_ITERATIONS

**Description:**

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

**Possible Values:**

> Any number between 0 and +inf.

**Default value:**

> 1000000

### 23.2.11   MSK_IPAR_CACHE_LICENSE

**Corresponding constant:**

> MSK_IPAR_CACHE_LICENSE

**Description:**

> Specifies if the license is kept checked out for the lifetime of the mosek environment (on) or returned to the server immediately after the optimization (off).
>
> Check-in and check-out of licenses have an overhead. Frequent communication with the license server should be avoided.

**Possible values:**

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

**Default value:**

> MSK_ON

### 23.2.12   MSK_IPAR_CHECK_CONVEXITY

**Corresponding constant:**

> MSK_IPAR_CHECK_CONVEXITY

**Description:**

> Specify the level of convexity check on quadratic problems

**Possible values:**

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

**Default value:**

> MSK_CHECK_CONVEXITY_FULL

## 23.2.13 MSK_IPAR_COMPRESS_STATFILE

**Corresponding constant:**

MSK_IPAR_COMPRESS_STATFILE

**Description:**

Control compression of stat files.

**Possible values:**

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

**Default value:**

MSK_ON

## 23.2.14 MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS

**Corresponding constant:**

MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS

**Description:**

The maximum number of simultaneous optimizations that will be started by the concurrent optimizer.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

2

## 23.2.15 MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX

**Corresponding constant:**

MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX

**Description:**

Priority of the dual simplex algorithm when selecting solvers for concurrent optimization.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

2

### 23.2.16   MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX

**Corresponding constant:**

MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX

**Description:**

Priority of the free simplex optimizer when selecting solvers for concurrent optimization.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

3

### 23.2.17   MSK_IPAR_CONCURRENT_PRIORITY_INTPNT

**Corresponding constant:**

MSK_IPAR_CONCURRENT_PRIORITY_INTPNT

**Description:**

Priority of the interior-point algorithm when selecting solvers for concurrent optimization.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

4

### 23.2.18   MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX

**Corresponding constant:**

MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX

**Description:**

Priority of the primal simplex algorithm when selecting solvers for concurrent optimization.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1

### 23.2.19 MSK_IPAR_FEASREPAIR_OPTIMIZE

**Corresponding constant:**

MSK_IPAR_FEASREPAIR_OPTIMIZE

**Description:**

Controls which type of feasibility analysis is to be performed.

**Possible values:**

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

- MSK_FEASREPAIR_OPTIMIZE_NONE Do not optimize the feasibility repair problem.

- MSK_FEASREPAIR_OPTIMIZE_PENALTY Minimize weighted sum of violations.

**Default value:**

MSK_FEASREPAIR_OPTIMIZE_NONE

### 23.2.20 MSK_IPAR_INFEAS_GENERIC_NAMES

**Corresponding constant:**

MSK_IPAR_INFEAS_GENERIC_NAMES

**Description:**

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

**Possible values:**

- MSK_OFF Switch the option off.

- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

### 23.2.21 MSK_IPAR_INFEAS_PREFER_PRIMAL

**Corresponding constant:**

MSK_IPAR_INFEAS_PREFER_PRIMAL

**Description:**

If both certificates of primal and dual infeasibility are supplied then only the primal is used when this option is turned on.

**Possible values:**

- MSK_OFF Switch the option off.

- MSK_ON Switch the option on.

**Default value:**

MSK_ON

### 23.2.22   MSK_IPAR_INFEAS_REPORT_AUTO

**Corresponding constant:**

MSK_IPAR_INFEAS_REPORT_AUTO

**Description:**

Controls whether an infeasibility report is automatically produced after the optimization if the problem is primal or dual infeasible.

**Possible values:**

- MSK_OFF Switch the option off.

- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

### 23.2.23   MSK_IPAR_INFEAS_REPORT_LEVEL

**Corresponding constant:**

MSK_IPAR_INFEAS_REPORT_LEVEL

**Description:**

Controls the amount of information presented in an infeasibility report. Higher values imply more information.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1

## 23.2.24 MSK_IPAR_INTPNT_BASIS

**Corresponding constant:**

MSK_IPAR_INTPNT_BASIS

**Description:**

Controls whether the interior-point optimizer also computes an optimal basis.

**Possible values:**

- MSK_BI_ALWAYS Basis identification is always performed even if the interior-point optimizer terminates abnormally.

- MSK_BI_IF_FEASIBLE Basis identification is not performed if the interior-point optimizer terminates with a problem status saying that the problem is primal or dual infeasible.

- MSK_BI_NEVER Never do basis identification.

- MSK_BI_NO_ERROR Basis identification is performed if the interior-point optimizer terminates without an error.

- MSK_BI_RESERVERED Not currently in use.

**Default value:**

MSK_BI_ALWAYS

**See also:**

- MSK_IPAR_BI_IGNORE_MAX_ITER Turns on basis identification in case the interior-point optimizer is terminated due to maximum number of iterations.

- MSK_IPAR_BI_IGNORE_NUM_ERROR Turns on basis identification in case the interior-point optimizer is terminated due to a numerical problem.

## 23.2.25 MSK_IPAR_INTPNT_DIFF_STEP

**Corresponding constant:**

MSK_IPAR_INTPNT_DIFF_STEP

**Description:**

Controls whether different step sizes are allowed in the primal and dual space.

**Possible values:**

- MSK_OFF Switch the option off.

- MSK_ON Switch the option on.

**Default value:**

MSK_ON

## 23.2.26    MSK_IPAR_INTPNT_FACTOR_DEBUG_LVL

**Corresponding constant:**

MSK_IPAR_INTPNT_FACTOR_DEBUG_LVL

**Description:**

Controls factorization debug level.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

0

## 23.2.27    MSK_IPAR_INTPNT_FACTOR_METHOD

**Corresponding constant:**

MSK_IPAR_INTPNT_FACTOR_METHOD

**Description:**

Controls the method used to factor the Newton equation system.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

0

## 23.2.28    MSK_IPAR_INTPNT_HOTSTART

**Corresponding constant:**

MSK_IPAR_INTPNT_HOTSTART

**Description:**

Currently not in use.

**Possible values:**

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

**Default value:**

MSK_INTPNT_HOTSTART_NONE

### 23.2.29 MSK_IPAR_INTPNT_MAX_ITERATIONS

**Corresponding constant:**

MSK_IPAR_INTPNT_MAX_ITERATIONS

**Description:**

Controls the maximum number of iterations allowed in the interior-point optimizer.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

400

### 23.2.30 MSK_IPAR_INTPNT_MAX_NUM_COR

**Corresponding constant:**

MSK_IPAR_INTPNT_MAX_NUM_COR

**Description:**

Controls the maximum number of correctors allowed by the multiple corrector procedure. A negative value means that MOSEK is making the choice.

**Possible Values:**

Any number between -1 and +inf.

**Default value:**

-1

### 23.2.31 MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS

**Corresponding constant:**

MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS

**Description:**

Maximum number of steps to be used by the iterative refinement of the search direction. A negative value implies that the optimizer Chooses the maximum number of iterative refinement steps.

**Possible Values:**

Any number between -inf and +inf.

**Default value:**

> -1

## 23.2.32   MSK_IPAR_INTPNT_OFF_COL_TRH

**Corresponding constant:**

> MSK_IPAR_INTPNT_OFF_COL_TRH

**Description:**

> Controls how many offending columns are detected in the Jacobian of the constraint matrix.
>
> 1 means aggressive detection, higher values mean less aggressive detection.
>
> 0 means no detection.

**Possible Values:**

> Any number between 0 and +inf.

**Default value:**

> 40

## 23.2.33   MSK_IPAR_INTPNT_ORDER_METHOD

**Corresponding constant:**

> MSK_IPAR_INTPNT_ORDER_METHOD

**Description:**

> Controls the ordering strategy used by the interior-point optimizer when factorizing the Newton equation system.

**Possible values:**

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

**Default value:**

> MSK_ORDER_METHOD_FREE

### 23.2.34   MSK_IPAR_INTPNT_REGULARIZATION_USE

**Corresponding constant:**

MSK_IPAR_INTPNT_REGULARIZATION_USE

**Description:**

Controls whether regularization is allowed.

**Possible values:**

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

**Default value:**

MSK_ON

### 23.2.35   MSK_IPAR_INTPNT_SCALING

**Corresponding constant:**

MSK_IPAR_INTPNT_SCALING

**Description:**

Controls how the problem is scaled before the interior-point optimizer is used.

**Possible values:**

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

**Default value:**

MSK_SCALING_FREE

### 23.2.36   MSK_IPAR_INTPNT_SOLVE_FORM

**Corresponding constant:**

MSK_IPAR_INTPNT_SOLVE_FORM

**Description:**

Controls whether the primal or the dual problem is solved.

**Possible values:**

- **MSK_SOLVE_DUAL** The optimizer should solve the dual problem.
- **MSK_SOLVE_FREE** The optimizer is free to solve either the primal or the dual problem.
- **MSK_SOLVE_PRIMAL** The optimizer should solve the primal problem.

**Default value:**

MSK_SOLVE_FREE

### 23.2.37   MSK_IPAR_INTPNT_STARTING_POINT

**Corresponding constant:**

MSK_IPAR_INTPNT_STARTING_POINT

**Description:**

Starting point used by the interior-point optimizer.

**Possible values:**

- **MSK_STARTING_POINT_CONSTANT** The optimizer constructs a starting point by assigning a constant value to all primal and dual variables. This starting point is normally robust.
- **MSK_STARTING_POINT_FREE** The starting point is chosen automatically.
- **MSK_STARTING_POINT_GUESS** The optimizer guesses a starting point.
- **MSK_STARTING_POINT_SATISFY_BOUNDS** The starting point is choosen to satisfy all the simple bounds on nonlinear variables. If this starting point is employed, then more care than usual should employed when choosing the bounds on the nonlinear variables. In particular very tight bounds should be avoided.

**Default value:**

MSK_STARTING_POINT_FREE

### 23.2.38   MSK_IPAR_LIC_TRH_EXPIRY_WRN

**Corresponding constant:**

MSK_IPAR_LIC_TRH_EXPIRY_WRN

**Description:**

If a license feature expires in a numbers days less than the value of this parameter then a warning will be issued.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

7

### 23.2.39   MSK_IPAR_LICENSE_DEBUG

**Corresponding constant:**

MSK_IPAR_LICENSE_DEBUG

**Description:**

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

**Possible values:**

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

**Default value:**

MSK_OFF

### 23.2.40   MSK_IPAR_LICENSE_PAUSE_TIME

**Corresponding constant:**

MSK_IPAR_LICENSE_PAUSE_TIME

**Description:**

If MSK_IPAR_LICENSE_WAIT=MSK_ON and no license is available, then MOSEK sleeps a number of milliseconds between each check of whether a license has become free.

**Possible Values:**

Any number between 0 and 1000000.

**Default value:**

100

### 23.2.41   MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS

**Corresponding constant:**

MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS

**Description:**

Controls whether license features expire warnings are suppressed.

**Possible values:**

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

**Default value:**

MSK_OFF

## 23.2.42   MSK_IPAR_LICENSE_WAIT

**Corresponding constant:**

MSK_IPAR_LICENSE_WAIT

**Description:**

If all licenses are in use MOSEK returns with an error code. However, by turning on this parameter MOSEK will wait for an available license.

**Possible values:**

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

**Default value:**

MSK_OFF

## 23.2.43   MSK_IPAR_LOG

**Corresponding constant:**

MSK_IPAR_LOG

**Description:**

Controls the amount of log information. The value 0 implies that all log information is suppressed. A higher level implies that more information is logged.

Please note that if a task is employed to solve a sequence of optimization problems the value of this parameter is reduced by the value of MSK_IPAR_LOG_CUT_SECOND_OPT for the second and any subsequent optimizations.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

10

**See also:**

- MSK_IPAR_LOG_CUT_SECOND_OPT Controls the reduction in the log levels for the second and any subsequent optimizations.

## 23.2.44   MSK_IPAR_LOG_BI

**Corresponding constant:**

MSK_IPAR_LOG_BI

**Description:**

Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

4

## 23.2.45   MSK_IPAR_LOG_BI_FREQ

**Corresponding constant:**

MSK_IPAR_LOG_BI_FREQ

**Description:**

Controls how frequent the optimizer outputs information about the basis identification and how frequent the user-defined call-back function is called.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

2500

## 23.2.46   MSK_IPAR_LOG_CHECK_CONVEXITY

**Corresponding constant:**

MSK_IPAR_LOG_CHECK_CONVEXITY

**Description:**

Controls logging in convexity check on quadratic problems. Set to a positive value to turn logging on.

If a quadratic coefficient matrix is found to violate the requirement of PSD (NSD) then a list of negative (positive) pivot elements is printed. The absolute value of the pivot elements is also shown.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

    0

## 23.2.47   MSK_IPAR_LOG_CONCURRENT

**Corresponding constant:**

    MSK_IPAR_LOG_CONCURRENT

**Description:**

Controls amount of output printed by the concurrent optimizer.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

    1

## 23.2.48   MSK_IPAR_LOG_CUT_SECOND_OPT

**Corresponding constant:**

    MSK_IPAR_LOG_CUT_SECOND_OPT

**Description:**

If a task is employed to solve a sequence of optimization problems, then the value of the log levels is reduced by the value of this parameter. E.g `MSK_IPAR_LOG` and `MSK_IPAR_LOG_SIM` are reduced by the value of this parameter for the second and any subsequent optimizations.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

    1

**See also:**

- `MSK_IPAR_LOG` Controls the amount of log information.
- `MSK_IPAR_LOG_INTPNT` Controls the amount of log information from the interior-point optimizers.
- `MSK_IPAR_LOG_MIO` Controls the amount of log information from the mixed-integer optimizers.
- `MSK_IPAR_LOG_SIM` Controls the amount of log information from the simplex optimizers.

## 23.2.49 MSK_IPAR_LOG_EXPAND

**Corresponding constant:**

MSK_IPAR_LOG_EXPAND

**Description:**

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

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

0

## 23.2.50 MSK_IPAR_LOG_FACTOR

**Corresponding constant:**

MSK_IPAR_LOG_FACTOR

**Description:**

If turned on, then the factor log lines are added to the log.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1

## 23.2.51 MSK_IPAR_LOG_FEAS_REPAIR

**Corresponding constant:**

MSK_IPAR_LOG_FEAS_REPAIR

**Description:**

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

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1

### 23.2.52   MSK_IPAR_LOG_FILE

**Corresponding constant:**

MSK_IPAR_LOG_FILE

**Description:**

If turned on, then some log info is printed when a file is written or read.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1

### 23.2.53   MSK_IPAR_LOG_HEAD

**Corresponding constant:**

MSK_IPAR_LOG_HEAD

**Description:**

If turned on, then a header line is added to the log.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1

### 23.2.54   MSK_IPAR_LOG_INFEAS_ANA

**Corresponding constant:**

MSK_IPAR_LOG_INFEAS_ANA

**Description:**

Controls amount of output printed by the infeasibility analyzer procedures. A higher level implies that more information is logged.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1

## 23.2.55 MSK_IPAR_LOG_INTPNT

**Corresponding constant:**

MSK_IPAR_LOG_INTPNT

**Description:**

Controls amount of output printed printed by the interior-point optimizer. A higher level implies that more information is logged.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

4

## 23.2.56 MSK_IPAR_LOG_MIO

**Corresponding constant:**

MSK_IPAR_LOG_MIO

**Description:**

Controls the log level for the mixed-integer optimizer. A higher level implies that more information is logged.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

4

## 23.2.57 MSK_IPAR_LOG_MIO_FREQ

**Corresponding constant:**

MSK_IPAR_LOG_MIO_FREQ

**Description:**

Controls how frequent the mixed-integer optimizer prints the log line. It will print line every time MSK_IPAR_LOG_MIO_FREQ relaxations have been solved.

**Possible Values:**

A integer value.

**Default value:**

1000

### 23.2.58   MSK_IPAR_LOG_NONCONVEX

**Corresponding constant:**

MSK_IPAR_LOG_NONCONVEX

**Description:**

Controls amount of output printed by the nonconvex optimizer.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1

### 23.2.59   MSK_IPAR_LOG_OPTIMIZER

**Corresponding constant:**

MSK_IPAR_LOG_OPTIMIZER

**Description:**

Controls the amount of general optimizer information that is logged.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1

### 23.2.60   MSK_IPAR_LOG_ORDER

**Corresponding constant:**

MSK_IPAR_LOG_ORDER

**Description:**

If turned on, then factor lines are added to the log.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1

### 23.2.61   MSK_IPAR_LOG_PARAM

**Corresponding constant:**

MSK_IPAR_LOG_PARAM

**Description:**

Controls the amount of information printed out about parameter changes.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

0

### 23.2.62   MSK_IPAR_LOG_PRESOLVE

**Corresponding constant:**

MSK_IPAR_LOG_PRESOLVE

**Description:**

Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1

### 23.2.63   MSK_IPAR_LOG_RESPONSE

**Corresponding constant:**

MSK_IPAR_LOG_RESPONSE

**Description:**

Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

0

## 23.2.64   MSK_IPAR_LOG_SENSITIVITY

**Corresponding constant:**

MSK_IPAR_LOG_SENSITIVITY

**Description:**

Controls the amount of logging during the sensitivity analysis. 0: Means no logging information is produced. 1: Timing information is printed. 2: Sensitivity results are printed.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1

## 23.2.65   MSK_IPAR_LOG_SENSITIVITY_OPT

**Corresponding constant:**

MSK_IPAR_LOG_SENSITIVITY_OPT

**Description:**

Controls the amount of logging from the optimizers employed during the sensitivity analysis. 0 means no logging information is produced.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

0

## 23.2.66   MSK_IPAR_LOG_SIM

**Corresponding constant:**

MSK_IPAR_LOG_SIM

**Description:**

Controls amount of output printed by the simplex optimizer. A higher level implies that more information is logged.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

4

### 23.2.67 MSK_IPAR_LOG_SIM_FREQ

**Corresponding constant:**

MSK_IPAR_LOG_SIM_FREQ

**Description:**

Controls how frequent the simplex optimizer outputs information about the optimization and how frequent the user-defined call-back function is called.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1000

### 23.2.68 MSK_IPAR_LOG_SIM_MINOR

**Corresponding constant:**

MSK_IPAR_LOG_SIM_MINOR

**Description:**

Currently not in use.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1

### 23.2.69 MSK_IPAR_LOG_SIM_NETWORK_FREQ

**Corresponding constant:**

MSK_IPAR_LOG_SIM_NETWORK_FREQ

**Description:**

Controls how frequent the network simplex optimizer outputs information about the optimization and how frequent the user-defined call-back function is called. The network optimizer will use a logging frequency equal to `MSK_IPAR_LOG_SIM_FREQ` times `MSK_IPAR_LOG_SIM_NETWORK_FREQ`.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1000

### 23.2.70   MSK_IPAR_LOG_STORAGE

**Corresponding constant:**

MSK_IPAR_LOG_STORAGE

**Description:**

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

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

0

### 23.2.71   MSK_IPAR_MAX_NUM_WARNINGS

**Corresponding constant:**

MSK_IPAR_MAX_NUM_WARNINGS

**Description:**

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

**Possible Values:**

Any number between -inf and +inf.

**Default value:**

6

### 23.2.72   MSK_IPAR_MIO_BRANCH_DIR

**Corresponding constant:**

MSK_IPAR_MIO_BRANCH_DIR

**Description:**

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

**Possible values:**

- MSK_BRANCH_DIR_DOWN The mixed-integer optimizer always chooses the down branch first.
- MSK_BRANCH_DIR_FREE The mixed-integer optimizer decides which branch to choose.
- MSK_BRANCH_DIR_UP The mixed-integer optimizer always chooses the up branch first.

**Default value:**

MSK_BRANCH_DIR_FREE

### 23.2.73   MSK_IPAR_MIO_BRANCH_PRIORITIES_USE

**Corresponding constant:**

MSK_IPAR_MIO_BRANCH_PRIORITIES_USE

**Description:**

Controls whether branching priorities are used by the mixed-integer optimizer.

**Possible values:**

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

**Default value:**

MSK_ON

### 23.2.74   MSK_IPAR_MIO_CONSTRUCT_SOL

**Corresponding constant:**

MSK_IPAR_MIO_CONSTRUCT_SOL

**Description:**

If set to MSK_ON and all integer variables have been given a value for which a feasible mixed integer solution exists, then MOSEK generates an initial solution to the mixed integer problem by fixing all integer values and solving the remaining problem.

**Possible values:**

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

**Default value:**

MSK_OFF

### 23.2.75   MSK_IPAR_MIO_CONT_SOL

**Corresponding constant:**

MSK_IPAR_MIO_CONT_SOL

**Description:**

Controls the meaning of the interior-point and basic solutions in mixed integer problems.

**Possible values:**

- **MSK_MIO_CONT_SOL_ITG** The reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. A solution is only reported in case the problem has a primal feasible solution.
- **MSK_MIO_CONT_SOL_ITG_REL** In case the problem is primal feasible then the reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. If the problem is primal infeasible, then the solution to the root node problem is reported.
- **MSK_MIO_CONT_SOL_NONE** No interior-point or basic solution are reported when the mixed-integer optimizer is used.
- **MSK_MIO_CONT_SOL_ROOT** The reported interior-point and basic solutions are a solution to the root node problem when mixed-integer optimizer is used.

**Default value:**

MSK_MIO_CONT_SOL_NONE

### 23.2.76    MSK_IPAR_MIO_CUT_CG

**Corresponding constant:**

MSK_IPAR_MIO_CUT_CG

**Description:**

Controls whether CG cuts should be generated.

**Possible values:**

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

**Default value:**

MSK_ON

### 23.2.77    MSK_IPAR_MIO_CUT_CMIR

**Corresponding constant:**

MSK_IPAR_MIO_CUT_CMIR

**Description:**

Controls whether mixed integer rounding cuts should be generated.

**Possible values:**

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

**Default value:**

MSK_ON

## 23.2.78 MSK_IPAR_MIO_CUT_LEVEL_ROOT

**Corresponding constant:**

MSK_IPAR_MIO_CUT_LEVEL_ROOT

**Description:**

Controls the cut level employed by the mixed-integer optimizer at the root node. A negative value means a default value determined by the mixed-integer optimizer is used. By adding the appropriate values from the following table the employed cut types can be controlled.

| | |
|---|---|
| GUB cover | +2 |
| Flow cover | +4 |
| Lifting | +8 |
| Plant location | +16 |
| Disaggregation | +32 |
| Knapsack cover | +64 |
| Lattice | +128 |
| Gomory | +256 |
| Coefficient reduction | +512 |
| GCD | +1024 |
| Obj. integrality | +2048 |

**Possible Values:**

Any value.

**Default value:**

-1

## 23.2.79 MSK_IPAR_MIO_CUT_LEVEL_TREE

**Corresponding constant:**

MSK_IPAR_MIO_CUT_LEVEL_TREE

**Description:**

Controls the cut level employed by the mixed-integer optimizer at the tree. See MSK_IPAR_MIO_CUT_LEVEL_ROOT for an explanation of the parameter values.

**Possible Values:**

Any value.

**Default value:**

-1

### 23.2.80   MSK_IPAR_MIO_FEASPUMP_LEVEL

**Corresponding constant:**

MSK_IPAR_MIO_FEASPUMP_LEVEL

**Description:**

Feasibility pump is a heuristic designed to compute an initial feasible solution. A value of 0 implies that the feasibility pump heuristic is not used. A value of -1 implies that the mixed-integer optimizer decides how the feasibility pump heuristic is used. A larger value than 1 implies that the feasibility pump is employed more aggressively. Normally a value beyond 3 is not worthwhile.

**Possible Values:**

Any number between -inf and 3.

**Default value:**

-1

### 23.2.81   MSK_IPAR_MIO_HEURISTIC_LEVEL

**Corresponding constant:**

MSK_IPAR_MIO_HEURISTIC_LEVEL

**Description:**

Controls the heuristic employed by the mixed-integer optimizer to locate an initial good integer feasible solution. A value of zero means the heuristic is not used at all. A larger value than 0 means that a gradually more sophisticated heuristic is used which is computationally more expensive. A negative value implies that the optimizer chooses the heuristic. Normally a value around 3 to 5 should be optimal.

**Possible Values:**

Any value.

**Default value:**

-1

### 23.2.82   MSK_IPAR_MIO_HOTSTART

**Corresponding constant:**

MSK_IPAR_MIO_HOTSTART

**Description:**

Controls whether the integer optimizer is hot-started.

**Possible values:**

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

**Default value:**

MSK_ON

### 23.2.83   MSK_IPAR_MIO_KEEP_BASIS

**Corresponding constant:**

MSK_IPAR_MIO_KEEP_BASIS

**Description:**

Controls whether the integer presolve keeps bases in memory. This speeds on the solution process at cost of bigger memory consumption.

**Possible values:**

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

**Default value:**

MSK_ON

### 23.2.84   MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER

**Corresponding constant:**

MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER

**Description:**

Controls the size of the local search space when doing local branching.

**Possible Values:**

Any number between -inf and +inf.

**Default value:**

-1

### 23.2.85   MSK_IPAR_MIO_MAX_NUM_BRANCHES

**Corresponding constant:**

MSK_IPAR_MIO_MAX_NUM_BRANCHES

**Description:**

>   Maximum number of branches allowed during the branch and bound search. A negative value means infinite.

**Possible Values:**

>   Any number between -inf and +inf.

**Default value:**

>   -1

**See also:**

>   - MSK_DPAR_MIO_DISABLE_TERM_TIME Certain termination criteria is disabled within the mixed-integer optimizer for period time specified by the parameter.

## 23.2.86    MSK_IPAR_MIO_MAX_NUM_RELAXS

**Corresponding constant:**

>   MSK_IPAR_MIO_MAX_NUM_RELAXS

**Description:**

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

**Possible Values:**

>   Any number between -inf and +inf.

**Default value:**

>   -1

**See also:**

>   - MSK_DPAR_MIO_DISABLE_TERM_TIME Certain termination criteria is disabled within the mixed-integer optimizer for period time specified by the parameter.

## 23.2.87    MSK_IPAR_MIO_MAX_NUM_SOLUTIONS

**Corresponding constant:**

>   MSK_IPAR_MIO_MAX_NUM_SOLUTIONS

**Description:**

>   The mixed-integer optimizer can be terminated after a certain number of different feasible solutions has been located. If this parameter has the value $n$ and $n$ is strictly positive, then the mixed-integer optimizer will be terminated when $n$ feasible solutions have been located.

**Possible Values:**

Any number between -inf and +inf.

**Default value:**

-1

**See also:**

- MSK_DPAR_MIO_DISABLE_TERM_TIME Certain termination criteria is disabled within the mixed-integer optimizer for period time specified by the parameter.

## 23.2.88   MSK_IPAR_MIO_MODE

**Corresponding constant:**

MSK_IPAR_MIO_MODE

**Description:**

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

**Possible values:**

- MSK_MIO_MODE_IGNORED The integer constraints are ignored and the problem is solved as a continuous problem.
- MSK_MIO_MODE_LAZY Integer restrictions should be satisfied if an optimizer is available for the problem.
- MSK_MIO_MODE_SATISFIED Integer restrictions should be satisfied.

**Default value:**

MSK_MIO_MODE_SATISFIED

## 23.2.89   MSK_IPAR_MIO_MT_USER_CB

**Corresponding constant:**

MSK_IPAR_MIO_MT_USER_CB

**Description:**

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

**Possible values:**

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

**Default value:**

MSK_ON

## 23.2.90   MSK_IPAR_MIO_NODE_OPTIMIZER

**Corresponding constant:**

MSK_IPAR_MIO_NODE_OPTIMIZER

**Description:**

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

**Possible values:**

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

**Default value:**

MSK_OPTIMIZER_FREE

## 23.2.91   MSK_IPAR_MIO_NODE_SELECTION

**Corresponding constant:**

MSK_IPAR_MIO_NODE_SELECTION

**Description:**

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

**Possible values:**

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

- MSK_MIO_NODE_SELECTION_HYBRID The optimizer employs a hybrid strategy.

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

- MSK_MIO_NODE_SELECTION_WORST The optimizer employs a worst bound node selection strategy.

**Default value:**

MSK_MIO_NODE_SELECTION_FREE

### 23.2.92    MSK_IPAR_MIO_OPTIMIZER_MODE

**Corresponding constant:**

MSK_IPAR_MIO_OPTIMIZER_MODE

**Description:**

An exprimental feature.

**Possible Values:**

Any number between 0 and 1.

**Default value:**

0

### 23.2.93    MSK_IPAR_MIO_PRESOLVE_AGGREGATE

**Corresponding constant:**

MSK_IPAR_MIO_PRESOLVE_AGGREGATE

**Description:**

Controls whether the presolve used by the mixed-integer optimizer tries to aggregate the constraints.

**Possible values:**

- MSK_OFF Switch the option off.

- MSK_ON Switch the option on.

**Default value:**

MSK_ON

## 23.2.94    MSK_IPAR_MIO_PRESOLVE_PROBING

**Corresponding constant:**

MSK_IPAR_MIO_PRESOLVE_PROBING

**Description:**

Controls whether the mixed-integer presolve performs probing. Probing can be very time consuming.

**Possible values:**

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

**Default value:**

MSK_ON

## 23.2.95    MSK_IPAR_MIO_PRESOLVE_USE

**Corresponding constant:**

MSK_IPAR_MIO_PRESOLVE_USE

**Description:**

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

**Possible values:**

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

**Default value:**

MSK_ON

## 23.2.96    MSK_IPAR_MIO_PROBING_LEVEL

**Corresponding constant:**

MSK_IPAR_MIO_PROBING_LEVEL

**Description:**

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

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

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

**Possible Values:**

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

**Default value:**

-1

## 23.2.97 MSK_IPAR_MIO_RINS_MAX_NODES

**Corresponding constant:**

MSK_IPAR_MIO_RINS_MAX_NODES

**Description:**

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

**Possible Values:**

Any number between -1 and +inf.

**Default value:**

-1

## 23.2.98 MSK_IPAR_MIO_ROOT_OPTIMIZER

**Corresponding constant:**

MSK_IPAR_MIO_ROOT_OPTIMIZER

**Description:**

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

**Possible values:**

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

- **MSK_OPTIMIZER_MIXED_INT_CONIC** The mixed-integer optimizer for conic and linear problems.

- **MSK_OPTIMIZER_NETWORK_PRIMAL_SIMPLEX** The network primal simplex optimizer is used. It is only applicable to pure network problems.

- **MSK_OPTIMIZER_NONCONVEX** The optimizer for nonconvex nonlinear problems.

- **MSK_OPTIMIZER_PRIMAL_DUAL_SIMPLEX** The primal dual simplex optimizer is used.

- **MSK_OPTIMIZER_PRIMAL_SIMPLEX** The primal simplex optimizer is used.

**Default value:**

MSK_OPTIMIZER_FREE

## 23.2.99   MSK_IPAR_MIO_STRONG_BRANCH

**Corresponding constant:**

MSK_IPAR_MIO_STRONG_BRANCH

**Description:**

The value specifies the depth from the root in which strong branching is used. A negative value means that the optimizer chooses a default value automatically.

**Possible Values:**

Any number between -inf and +inf.

**Default value:**

-1

## 23.2.100   MSK_IPAR_MIO_USE_MULTITHREADED_OPTIMIZER

**Corresponding constant:**

MSK_IPAR_MIO_USE_MULTITHREADED_OPTIMIZER

**Description:**

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

**Possible values:**

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

**Default value:**

MSK_OFF

### 23.2.101 MSK_IPAR_MT_SPINCOUNT

**Corresponding constant:**

MSK_IPAR_MT_SPINCOUNT

**Description:**

Set the number of iterations to spin before sleeping.

**Possible Values:**

Any integer greater or equal to 0.

**Default value:**

0

### 23.2.102 MSK_IPAR_NONCONVEX_MAX_ITERATIONS

**Corresponding constant:**

MSK_IPAR_NONCONVEX_MAX_ITERATIONS

**Description:**

Maximum number of iterations that can be used by the nonconvex optimizer.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

100000

### 23.2.103 MSK_IPAR_NUM_THREADS

**Corresponding constant:**

MSK_IPAR_NUM_THREADS

**Description:**

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

**Possible Values:**

Any integer greater or equal to 0.

**Default value:**

0

## 23.2.104    MSK_IPAR_OPF_MAX_TERMS_PER_LINE

**Corresponding constant:**

MSK_IPAR_OPF_MAX_TERMS_PER_LINE

**Description:**

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

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

5

## 23.2.105    MSK_IPAR_OPF_WRITE_HEADER

**Corresponding constant:**

MSK_IPAR_OPF_WRITE_HEADER

**Description:**

Write a text header with date and MOSEK version in an OPF file.

**Possible values:**

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

**Default value:**

MSK_ON

## 23.2.106    MSK_IPAR_OPF_WRITE_HINTS

**Corresponding constant:**

MSK_IPAR_OPF_WRITE_HINTS

**Description:**

Write a hint section with problem dimensions in the beginning of an OPF file.

**Possible values:**

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

**Default value:**

MSK_ON

## 23.2.107 MSK_IPAR_OPF_WRITE_PARAMETERS

**Corresponding constant:**

MSK_IPAR_OPF_WRITE_PARAMETERS

**Description:**

Write a parameter section in an OPF file.

**Possible values:**

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

**Default value:**

MSK_OFF

## 23.2.108 MSK_IPAR_OPF_WRITE_PROBLEM

**Corresponding constant:**

MSK_IPAR_OPF_WRITE_PROBLEM

**Description:**

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

**Possible values:**

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

**Default value:**

MSK_ON

## 23.2.109 MSK_IPAR_OPF_WRITE_SOL_BAS

**Corresponding constant:**

MSK_IPAR_OPF_WRITE_SOL_BAS

**Description:**

If MSK_IPAR_OPF_WRITE_SOLUTIONS is MSK_ON and a basic solution is defined, include the basic solution in OPF files.

**Possible values:**

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

**Default value:**

MSK_ON

### 23.2.110    MSK_IPAR_OPF_WRITE_SOL_ITG

**Corresponding constant:**

MSK_IPAR_OPF_WRITE_SOL_ITG

**Description:**

If MSK_IPAR_OPF_WRITE_SOLUTIONS is MSK_ON and an integer solution is defined, write the integer solution in OPF files.

**Possible values:**

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

**Default value:**

MSK_ON

### 23.2.111    MSK_IPAR_OPF_WRITE_SOL_ITR

**Corresponding constant:**

MSK_IPAR_OPF_WRITE_SOL_ITR

**Description:**

If MSK_IPAR_OPF_WRITE_SOLUTIONS is MSK_ON and an interior solution is defined, write the interior solution in OPF files.

**Possible values:**

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

**Default value:**

MSK_ON

### 23.2.112    MSK_IPAR_OPF_WRITE_SOLUTIONS

**Corresponding constant:**

MSK_IPAR_OPF_WRITE_SOLUTIONS

**Description:**

Enable inclusion of solutions in the OPF files.

**Possible values:**

- MSK_OFF Switch the option off.

- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

## 23.2.113 MSK_IPAR_OPTIMIZER

**Corresponding constant:**

MSK_IPAR_OPTIMIZER

**Description:**

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

**Possible values:**

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

**Default value:**

MSK_OPTIMIZER_FREE

## 23.2.114 MSK_IPAR_PARAM_READ_CASE_NAME

**Corresponding constant:**

MSK_IPAR_PARAM_READ_CASE_NAME

**Description:**

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

**Possible values:**

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

**Default value:**

`MSK_ON`

### 23.2.115   MSK_IPAR_PARAM_READ_IGN_ERROR

**Corresponding constant:**

MSK_IPAR_PARAM_READ_IGN_ERROR

**Description:**

If turned on, then errors in paramter settings is ignored.

**Possible values:**

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

**Default value:**

`MSK_OFF`

### 23.2.116   MSK_IPAR_PRESOLVE_ELIM_FILL

**Corresponding constant:**

MSK_IPAR_PRESOLVE_ELIM_FILL

**Description:**

Controls the maximum amount of fill-in that can be created during the elimination phase of the presolve. This parameter times (`numcon`+`numvar`) denotes the amount of fill-in.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1

### 23.2.117 MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES

**Corresponding constant:**

MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES

**Description:**

Control the maximum number of times the eliminator is tried.

**Possible Values:**

A negative value implies MOSEK decides maximum number of times.

**Default value:**

-1

### 23.2.118 MSK_IPAR_PRESOLVE_ELIMINATOR_USE

**Corresponding constant:**

MSK_IPAR_PRESOLVE_ELIMINATOR_USE

**Description:**

Controls whether free or implied free variables are eliminated from the problem.

**Possible values:**

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

**Default value:**

MSK_ON

### 23.2.119 MSK_IPAR_PRESOLVE_LEVEL

**Corresponding constant:**

MSK_IPAR_PRESOLVE_LEVEL

**Description:**

Currently not used.

**Possible Values:**

Any number between -inf and +inf.

**Default value:**

-1

### 23.2.120   MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH

**Corresponding constant:**

MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH

**Description:**

The linear dependency check is potentially computationally expensive.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

100

### 23.2.121   MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH

**Corresponding constant:**

MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH

**Description:**

The linear dependency check is potentially computationally expensive.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

100

### 23.2.122   MSK_IPAR_PRESOLVE_LINDEP_USE

**Corresponding constant:**

MSK_IPAR_PRESOLVE_LINDEP_USE

**Description:**

Controls whether the linear constraints are checked for linear dependencies.

**Possible values:**

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

**Default value:**

MSK_ON

### 23.2.123    MSK_IPAR_PRESOLVE_MAX_NUM_REDUCTIONS

**Corresponding constant:**

MSK_IPAR_PRESOLVE_MAX_NUM_REDUCTIONS

**Description:**

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

**Possible Values:**

Any number between -inf and +inf.

**Default value:**

-1

### 23.2.124    MSK_IPAR_PRESOLVE_USE

**Corresponding constant:**

MSK_IPAR_PRESOLVE_USE

**Description:**

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

**Possible values:**

- MSK_PRESOLVE_MODE_FREE It is decided automatically whether to presolve before the problem is optimized.
- MSK_PRESOLVE_MODE_OFF The problem is not presolved before it is optimized.
- MSK_PRESOLVE_MODE_ON The problem is presolved before it is optimized.

**Default value:**

MSK_PRESOLVE_MODE_FREE

### 23.2.125    MSK_IPAR_PRIMAL_REPAIR_OPTIMIZER

**Corresponding constant:**

MSK_IPAR_PRIMAL_REPAIR_OPTIMIZER

**Description:**

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

**Possible values:**

- MSK_OPTIMIZER_CONCURRENT The optimizer for nonconvex nonlinear problems.

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

**Default value:**

MSK_OPTIMIZER_FREE

### 23.2.126    MSK_IPAR_QO_SEPARABLE_REFORMULATION

**Corresponding constant:**

MSK_IPAR_QO_SEPARABLE_REFORMULATION

**Description:**

Determine if Quadratic programing problems should be reformulated to separable form.

**Possible values:**

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

**Default value:**

MSK_OFF

### 23.2.127    MSK_IPAR_READ_ANZ

**Corresponding constant:**

MSK_IPAR_READ_ANZ

**Description:**

Expected maximum number of $A$ non-zeros to be read. The option is used only by fast MPS and LP file readers.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

100000

### 23.2.128  MSK_IPAR_READ_CON

**Corresponding constant:**

MSK_IPAR_READ_CON

**Description:**

Expected maximum number of constraints to be read. The option is only used by fast MPS and LP file readers.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

10000

### 23.2.129  MSK_IPAR_READ_CONE

**Corresponding constant:**

MSK_IPAR_READ_CONE

**Description:**

Expected maximum number of conic constraints to be read. The option is used only by fast MPS and LP file readers.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

2500

### 23.2.130  MSK_IPAR_READ_DATA_COMPRESSED

**Corresponding constant:**

MSK_IPAR_READ_DATA_COMPRESSED

**Description:**

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

**Possible values:**

- MSK_COMPRESS_FREE The type of compression used is chosen automatically.
- MSK_COMPRESS_GZIP The type of compression used is gzip compatible.
- MSK_COMPRESS_NONE No compression is used.

**Default value:**

MSK_COMPRESS_FREE

### 23.2.131   MSK_IPAR_READ_DATA_FORMAT

**Corresponding constant:**

MSK_IPAR_READ_DATA_FORMAT

**Description:**

Format of the data file to be read.

**Possible values:**

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

**Default value:**

MSK_DATA_FORMAT_EXTENSION

### 23.2.132   MSK_IPAR_READ_DEBUG

**Corresponding constant:**

MSK_IPAR_READ_DEBUG

**Description:**

Turns on additional debugging information when reading files.

**Possible values:**

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

**Default value:**

MSK_OFF

### 23.2.133 MSK_IPAR_READ_KEEP_FREE_CON

**Corresponding constant:**

MSK_IPAR_READ_KEEP_FREE_CON

**Description:**

Controls whether the free constraints are included in the problem.

**Possible values:**

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

**Default value:**

MSK_OFF

### 23.2.134 MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU

**Corresponding constant:**

MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU

**Description:**

If this option is turned on, MOSEK will drop variables that are defined for the first time in the bounds section.

**Possible values:**

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

**Default value:**

MSK_OFF

### 23.2.135 MSK_IPAR_READ_LP_QUOTED_NAMES

**Corresponding constant:**

MSK_IPAR_READ_LP_QUOTED_NAMES

**Description:**

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

**Possible values:**

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

**Default value:**

MSK_ON

### 23.2.136    MSK IPAR READ MPS FORMAT

**Corresponding constant:**

MSK_IPAR_READ_MPS_FORMAT

**Description:**

Controls how strictly the MPS file reader interprets the MPS format.

**Possible values:**

- MSK_MPS_FORMAT_FREE It is assumed that the input file satisfies the free MPS format. This implies that spaces are not allowed in names. Otherwise the format is free.
- MSK_MPS_FORMAT_RELAXED It is assumed that the input file satisfies a slightly relaxed version of the MPS format.
- MSK_MPS_FORMAT_STRICT It is assumed that the input file satisfies the MPS format strictly.

**Default value:**

MSK_MPS_FORMAT_RELAXED

### 23.2.137    MSK IPAR READ MPS KEEP INT

**Corresponding constant:**

MSK_IPAR_READ_MPS_KEEP_INT

**Description:**

Controls whether MOSEK should keep the integer restrictions on the variables while reading the MPS file.

**Possible values:**

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

**Default value:**

MSK_ON

### 23.2.138    MSK IPAR READ MPS OBJ SENSE

**Corresponding constant:**

MSK_IPAR_READ_MPS_OBJ_SENSE

**Description:**

If turned on, the MPS reader uses the objective sense section. Otherwise the MPS reader ignores it.

**Possible values:**

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

**Default value:**

MSK_ON

## 23.2.139 MSK_IPAR_READ_MPS_RELAX

**Corresponding constant:**

MSK_IPAR_READ_MPS_RELAX

**Description:**

If this option is turned on, then mixed integer constraints are ignored when a problem is read.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_ON

## 23.2.140 MSK_IPAR_READ_MPS_WIDTH

**Corresponding constant:**

MSK_IPAR_READ_MPS_WIDTH

**Description:**

Controls the maximal number of characters allowed in one line of the MPS file.

**Possible Values:**

Any positive number greater than 80.

**Default value:**

1024

### 23.2.141 MSK_IPAR_READ_QNZ

**Corresponding constant:**

MSK_IPAR_READ_QNZ

**Description:**

Expected maximum number of $Q$ non-zeros to be read. The option is used only by MPS and LP file readers.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

20000

### 23.2.142 MSK_IPAR_READ_TASK_IGNORE_PARAM

**Corresponding constant:**

MSK_IPAR_READ_TASK_IGNORE_PARAM

**Description:**

Controls whether MOSEK should ignore the parameter setting defined in the task file and use the default parameter setting instead.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

### 23.2.143 MSK_IPAR_READ_VAR

**Corresponding constant:**

MSK_IPAR_READ_VAR

**Description:**

Expected maximum number of variable to be read. The option is used only by MPS and LP file readers.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

10000

### 23.2.144  MSK_IPAR_SENSITIVITY_ALL

**Corresponding constant:**

MSK_IPAR_SENSITIVITY_ALL

**Description:**

Not applicable.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

### 23.2.145  MSK_IPAR_SENSITIVITY_OPTIMIZER

**Corresponding constant:**

MSK_IPAR_SENSITIVITY_OPTIMIZER

**Description:**

Controls which optimizer is used for optimal partition sensitivity analysis.

**Possible values:**

- MSK_OPTIMIZER_CONCURRENT The optimizer for nonconvex nonlinear problems.
- MSK_OPTIMIZER_CONIC The optimizer for problems having conic constraints.
- MSK_OPTIMIZER_DUAL_SIMPLEX The dual simplex optimizer is used.
- MSK_OPTIMIZER_FREE The optimizer is chosen automatically.
- MSK_OPTIMIZER_FREE_SIMPLEX One of the simplex optimizers is used.
- MSK_OPTIMIZER_INTPNT The interior-point optimizer is used.
- MSK_OPTIMIZER_MIXED_INT The mixed-integer optimizer.
- MSK_OPTIMIZER_MIXED_INT_CONIC The mixed-integer optimizer for conic and linear problems.
- MSK_OPTIMIZER_NETWORK_PRIMAL_SIMPLEX The network primal simplex optimizer is used. It is only applicable to pure network problems.
- MSK_OPTIMIZER_NONCONVEX The optimizer for nonconvex nonlinear problems.
- MSK_OPTIMIZER_PRIMAL_DUAL_SIMPLEX The primal dual simplex optimizer is used.
- MSK_OPTIMIZER_PRIMAL_SIMPLEX The primal simplex optimizer is used.

**Default value:**

MSK_OPTIMIZER_FREE_SIMPLEX

## 23.2.146    MSK_IPAR_SENSITIVITY_TYPE

**Corresponding constant:**

MSK_IPAR_SENSITIVITY_TYPE

**Description:**

Controls which type of sensitivity analysis is to be performed.

**Possible values:**

- MSK_SENSITIVITY_TYPE_BASIS Basis sensitivity analysis is performed.
- MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION Optimal partition sensitivity analysis is performed.

**Default value:**

MSK_SENSITIVITY_TYPE_BASIS

## 23.2.147    MSK_IPAR_SIM_BASIS_FACTOR_USE

**Corresponding constant:**

MSK_IPAR_SIM_BASIS_FACTOR_USE

**Description:**

Controls whether a (LU) factorization of the basis is used in a hot-start. Forcing a refactorization sometimes improves the stability of the simplex optimizers, but in most cases there is a performance penanlty.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_ON

## 23.2.148    MSK_IPAR_SIM_DEGEN

**Corresponding constant:**

MSK_IPAR_SIM_DEGEN

**Description:**

Controls how aggressively degeneration is handled.

**Possible values:**

- **MSK_SIM_DEGEN_AGGRESSIVE** The simplex optimizer should use an aggressive degeneration strategy.

- **MSK_SIM_DEGEN_FREE** The simplex optimizer chooses the degeneration strategy.

- **MSK_SIM_DEGEN_MINIMUM** The simplex optimizer should use a minimum degeneration strategy.

- **MSK_SIM_DEGEN_MODERATE** The simplex optimizer should use a moderate degeneration strategy.

- **MSK_SIM_DEGEN_NONE** The simplex optimizer should use no degeneration strategy.

**Default value:**

MSK_SIM_DEGEN_FREE

## 23.2.149 MSK_IPAR_SIM_DUAL_CRASH

**Corresponding constant:**

MSK_IPAR_SIM_DUAL_CRASH

**Description:**

Controls whether crashing is performed in the dual simplex optimizer.

In general if a basis consists of more than (100-this parameter value)% fixed variables, then a crash will be performed.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

90

## 23.2.150 MSK_IPAR_SIM_DUAL_PHASEONE_METHOD

**Corresponding constant:**

MSK_IPAR_SIM_DUAL_PHASEONE_METHOD

**Description:**

An exprimental feature.

**Possible Values:**

Any number between 0 and 10.

**Default value:**

0

### 23.2.151    MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION

**Corresponding constant:**

MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION

**Description:**

The dual simplex optimizer can use a so-called restricted selection/pricing strategy to chooses the outgoing variable. Hence, if restricted selection is applied, then the dual simplex optimizer first choose a subset of all the potential outgoing variables. Next, for some time it will choose the outgoing variable only among the subset. From time to time the subset is redefined.

A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.

**Possible Values:**

Any number between 0 and 100.

**Default value:**

50

### 23.2.152    MSK_IPAR_SIM_DUAL_SELECTION

**Corresponding constant:**

MSK_IPAR_SIM_DUAL_SELECTION

**Description:**

Controls the choice of the incoming variable, known as the selection strategy, in the dual simplex optimizer.

**Possible values:**

- MSK_SIM_SELECTION_ASE The optimizer uses approximate steepest-edge pricing.
- MSK_SIM_SELECTION_DEVEX The optimizer uses devex steepest-edge pricing (or if it is not available an approximate steep-edge selection).
- MSK_SIM_SELECTION_FREE The optimizer chooses the pricing strategy.
- MSK_SIM_SELECTION_FULL The optimizer uses full pricing.
- MSK_SIM_SELECTION_PARTIAL The optimizer uses a partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.
- MSK_SIM_SELECTION_SE The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

**Default value:**

MSK_SIM_SELECTION_FREE

### 23.2.153   MSK_IPAR_SIM_EXPLOIT_DUPVEC

**Corresponding constant:**

MSK_IPAR_SIM_EXPLOIT_DUPVEC

**Description:**

Controls if the simplex optimizers are allowed to exploit duplicated columns.

**Possible values:**

- MSK_SIM_EXPLOIT_DUPVEC_FREE The simplex optimizer can choose freely.
- MSK_SIM_EXPLOIT_DUPVEC_OFF Disallow the simplex optimizer to exploit duplicated columns.
- MSK_SIM_EXPLOIT_DUPVEC_ON Allow the simplex optimizer to exploit duplicated columns.

**Default value:**

MSK_SIM_EXPLOIT_DUPVEC_OFF

### 23.2.154   MSK_IPAR_SIM_HOTSTART

**Corresponding constant:**

MSK_IPAR_SIM_HOTSTART

**Description:**

Controls the type of hot-start that the simplex optimizer perform.

**Possible values:**

- MSK_SIM_HOTSTART_FREE The simplex optimize chooses the hot-start type.
- MSK_SIM_HOTSTART_NONE The simplex optimizer performs a coldstart.
- MSK_SIM_HOTSTART_STATUS_KEYS Only the status keys of the constraints and variables are used to choose the type of hot-start.

**Default value:**

MSK_SIM_HOTSTART_FREE

### 23.2.155   MSK_IPAR_SIM_HOTSTART_LU

**Corresponding constant:**

MSK_IPAR_SIM_HOTSTART_LU

**Description:**

Determines if the simplex optimizer should exploit the initial factorization.

**Possible values:**

- **MSK_OFF** Switch the option off.
- **MSK_ON** Switch the option on.

**Default value:**

MSK_ON

### 23.2.156    MSK_IPAR_SIM_INTEGER

**Corresponding constant:**

MSK_IPAR_SIM_INTEGER

**Description:**

An exprimental feature.

**Possible Values:**

Any number between 0 and 10.

**Default value:**

0

### 23.2.157    MSK_IPAR_SIM_MAX_ITERATIONS

**Corresponding constant:**

MSK_IPAR_SIM_MAX_ITERATIONS

**Description:**

Maximum number of iterations that can be used by a simplex optimizer.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

10000000

### 23.2.158    MSK_IPAR_SIM_MAX_NUM_SETBACKS

**Corresponding constant:**

MSK_IPAR_SIM_MAX_NUM_SETBACKS

**Description:**

Controls how many set-backs are allowed within a simplex optimizer. A set-back is an event where the optimizer moves in the wrong direction. This is impossible in theory but may happen due to numerical problems.

**Possible Values:**

    Any number between 0 and +inf.

**Default value:**

    250

### 23.2.159 MSK_IPAR_SIM_NON_SINGULAR

**Corresponding constant:**

    MSK_IPAR_SIM_NON_SINGULAR

**Description:**

    Controls if the simplex optimizer ensures a non-singular basis, if possible.

**Possible values:**

- **MSK_OFF** Switch the option off.
- **MSK_ON** Switch the option on.

**Default value:**

    MSK_ON

### 23.2.160 MSK_IPAR_SIM_PRIMAL_CRASH

**Corresponding constant:**

    MSK_IPAR_SIM_PRIMAL_CRASH

**Description:**

    Controls whether crashing is performed in the primal simplex optimizer.

    In general, if a basis consists of more than (100-this parameter value)% fixed variables, then a crash will be performed.

**Possible Values:**

    Any nonnegative integer value.

**Default value:**

    90

### 23.2.161    MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD

**Corresponding constant:**

MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD

**Description:**

An exprimental feature.

**Possible Values:**

Any number between 0 and 10.

**Default value:**

0

### 23.2.162    MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION

**Corresponding constant:**

MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION

**Description:**

The primal simplex optimizer can use a so-called restricted selection/pricing strategy to chooses the outgoing variable. Hence, if restricted selection is applied, then the primal simplex optimizer first choose a subset of all the potential incoming variables. Next, for some time it will choose the incoming variable only among the subset. From time to time the subset is redefined.

A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.

**Possible Values:**

Any number between 0 and 100.

**Default value:**

50

### 23.2.163    MSK_IPAR_SIM_PRIMAL_SELECTION

**Corresponding constant:**

MSK_IPAR_SIM_PRIMAL_SELECTION

**Description:**

Controls the choice of the incoming variable, known as the selection strategy, in the primal simplex optimizer.

**Possible values:**

- **MSK_SIM_SELECTION_ASE** The optimizer uses approximate steepest-edge pricing.

- MSK_SIM_SELECTION_DEVEX The optimizer uses devex steepest-edge pricing (or if it is not available an approximate steep-edge selection).

- MSK_SIM_SELECTION_FREE The optimizer chooses the pricing strategy.

- MSK_SIM_SELECTION_FULL The optimizer uses full pricing.

- MSK_SIM_SELECTION_PARTIAL The optimizer uses a partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.

- MSK_SIM_SELECTION_SE The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

**Default value:**

MSK_SIM_SELECTION_FREE

### 23.2.164 MSK_IPAR_SIM_REFACTOR_FREQ

**Corresponding constant:**

MSK_IPAR_SIM_REFACTOR_FREQ

**Description:**

Controls how frequent the basis is refactorized. The value 0 means that the optimizer determines the best point of refactorization.

It is strongly recommended NOT to change this parameter.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

0

### 23.2.165 MSK_IPAR_SIM_REFORMULATION

**Corresponding constant:**

MSK_IPAR_SIM_REFORMULATION

**Description:**

Controls if the simplex optimizers are allowed to reformulate the problem.

**Possible values:**

- MSK_SIM_REFORMULATION_AGGRESSIVE The simplex optimizer should use an aggressive reformulation strategy.

- MSK_SIM_REFORMULATION_FREE The simplex optimizer can choose freely.

- MSK_SIM_REFORMULATION_OFF Disallow the simplex optimizer to reformulate the problem.

- MSK_SIM_REFORMULATION_ON Allow the simplex optimizer to reformulate the problem.

**Default value:**

MSK_SIM_REFORMULATION_OFF

## 23.2.166 MSK_IPAR_SIM_SAVE_LU

**Corresponding constant:**

MSK_IPAR_SIM_SAVE_LU

**Description:**

Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

## 23.2.167 MSK_IPAR_SIM_SCALING

**Corresponding constant:**

MSK_IPAR_SIM_SCALING

**Description:**

Controls how much effort is used in scaling the problem before a simplex optimizer is used.

**Possible values:**

- MSK_SCALING_AGGRESSIVE A very aggressive scaling is performed.
- MSK_SCALING_FREE The optimizer chooses the scaling heuristic.
- MSK_SCALING_MODERATE A conservative scaling is performed.
- MSK_SCALING_NONE No scaling is performed.

**Default value:**

MSK_SCALING_FREE

### 23.2.168 MSK_IPAR_SIM_SCALING_METHOD

**Corresponding constant:**

MSK_IPAR_SIM_SCALING_METHOD

**Description:**

Controls how the problem is scaled before a simplex optimizer is used.

**Possible values:**

- MSK_SCALING_METHOD_FREE The optimizer chooses the scaling heuristic.
- MSK_SCALING_METHOD_POW2 Scales only with power of 2 leaving the mantissa untouched.

**Default value:**

MSK_SCALING_METHOD_POW2

### 23.2.169 MSK_IPAR_SIM_SOLVE_FORM

**Corresponding constant:**

MSK_IPAR_SIM_SOLVE_FORM

**Description:**

Controls whether the primal or the dual problem is solved by the primal-/dual- simplex optimizer.

**Possible values:**

- MSK_SOLVE_DUAL The optimizer should solve the dual problem.
- MSK_SOLVE_FREE The optimizer is free to solve either the primal or the dual problem.
- MSK_SOLVE_PRIMAL The optimizer should solve the primal problem.

**Default value:**

MSK_SOLVE_FREE

### 23.2.170 MSK_IPAR_SIM_STABILITY_PRIORITY

**Corresponding constant:**

MSK_IPAR_SIM_STABILITY_PRIORITY

**Description:**

Controls how high priority the numerical stability should be given.

**Possible Values:**

Any number between 0 and 100.

**Default value:**

50

### 23.2.171    MSK_IPAR_SIM_SWITCH_OPTIMIZER

**Corresponding constant:**

MSK_IPAR_SIM_SWITCH_OPTIMIZER

**Description:**

The simplex optimizer sometimes chooses to solve the dual problem instead of the primal problem. This implies that if you have chosen to use the dual simplex optimizer and the problem is dualized, then it actually makes sense to use the primal simplex optimizer instead. If this parameter is on and the problem is dualized and furthermore the simplex optimizer is chosen to be the primal (dual) one, then it is switched to the dual (primal).

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

### 23.2.172    MSK_IPAR_SOL_FILTER_KEEP_BASIC

**Corresponding constant:**

MSK_IPAR_SOL_FILTER_KEEP_BASIC

**Description:**

If turned on, then basic and super basic constraints and variables are written to the solution file independent of the filter setting.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

### 23.2.173    MSK_IPAR_SOL_FILTER_KEEP_RANGED

**Corresponding constant:**

MSK_IPAR_SOL_FILTER_KEEP_RANGED

**Description:**

If turned on, then ranged constraints and variables are written to the solution file independent of the filter setting.

**Possible values:**

- **MSK_OFF** Switch the option off.
- **MSK_ON** Switch the option on.

**Default value:**

MSK_OFF

### 23.2.174   MSK_IPAR_SOL_READ_NAME_WIDTH

**Corresponding constant:**

MSK_IPAR_SOL_READ_NAME_WIDTH

**Description:**

When a solution is read by MOSEK and some constraint, variable or cone names contain blanks, then a maximum name width much be specified. A negative value implies that no name contain blanks.

**Possible Values:**

Any number between -inf and +inf.

**Default value:**

-1

### 23.2.175   MSK_IPAR_SOL_READ_WIDTH

**Corresponding constant:**

MSK_IPAR_SOL_READ_WIDTH

**Description:**

Controls the maximal acceptable width of line in the solutions when read by MOSEK.

**Possible Values:**

Any positive number greater than 80.

**Default value:**

1024

## 23.2.176    MSK_IPAR_SOLUTION_CALLBACK

**Corresponding constant:**

MSK_IPAR_SOLUTION_CALLBACK

**Description:**

Indicates whether solution call-backs will be performed during the optimization.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

## 23.2.177    MSK_IPAR_TIMING_LEVEL

**Corresponding constant:**

MSK_IPAR_TIMING_LEVEL

**Description:**

Controls the a amount of timing performed inside MOSEK.

**Possible Values:**

Any integer greater or equal to 0.

**Default value:**

1

## 23.2.178    MSK_IPAR_WARNING_LEVEL

**Corresponding constant:**

MSK_IPAR_WARNING_LEVEL

**Description:**

Deprecated and not in use

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1

### 23.2.179    MSK_IPAR_WRITE_BAS_CONSTRAINTS

**Corresponding constant:**

MSK_IPAR_WRITE_BAS_CONSTRAINTS

**Description:**

Controls whether the constraint section is written to the basic solution file.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_ON

### 23.2.180    MSK_IPAR_WRITE_BAS_HEAD

**Corresponding constant:**

MSK_IPAR_WRITE_BAS_HEAD

**Description:**

Controls whether the header section is written to the basic solution file.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_ON

### 23.2.181    MSK_IPAR_WRITE_BAS_VARIABLES

**Corresponding constant:**

MSK_IPAR_WRITE_BAS_VARIABLES

**Description:**

Controls whether the variables section is written to the basic solution file.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_ON

## 23.2.182   MSK_IPAR_WRITE_DATA_COMPRESSED

**Corresponding constant:**

MSK_IPAR_WRITE_DATA_COMPRESSED

**Description:**

Controls whether the data file is compressed while it is written. 0 means no compression while higher values mean more compression.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

0

## 23.2.183   MSK_IPAR_WRITE_DATA_FORMAT

**Corresponding constant:**

MSK_IPAR_WRITE_DATA_FORMAT

**Description:**

Controls the file format when writing task data to a file.

**Possible values:**

- MSK_DATA_FORMAT_CB Conic benchmark format.
- MSK_DATA_FORMAT_EXTENSION The file extension is used to determine the data file format.
- MSK_DATA_FORMAT_FREE_MPS The data data a free MPS formatted file.
- MSK_DATA_FORMAT_LP The data file is LP formatted.
- MSK_DATA_FORMAT_MPS The data file is MPS formatted.
- MSK_DATA_FORMAT_OP The data file is an optimization problem formatted file.
- MSK_DATA_FORMAT_TASK Generic task dump file.
- MSK_DATA_FORMAT_XML The data file is an XML formatted file.

**Default value:**

MSK_DATA_FORMAT_EXTENSION

### 23.2.184 MSK_IPAR_WRITE_DATA_PARAM

**Corresponding constant:**

MSK_IPAR_WRITE_DATA_PARAM

**Description:**

If this option is turned on the parameter settings are written to the data file as parameters.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

### 23.2.185 MSK_IPAR_WRITE_FREE_CON

**Corresponding constant:**

MSK_IPAR_WRITE_FREE_CON

**Description:**

Controls whether the free constraints are written to the data file.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

### 23.2.186 MSK_IPAR_WRITE_GENERIC_NAMES

**Corresponding constant:**

MSK_IPAR_WRITE_GENERIC_NAMES

**Description:**

Controls whether the generic names or user-defined names are used in the data file.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

### 23.2.187    MSK_IPAR_WRITE_GENERIC_NAMES_IO

**Corresponding constant:**

MSK_IPAR_WRITE_GENERIC_NAMES_IO

**Description:**

Index origin used in generic names.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

1

### 23.2.188    MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_CONIC_ITEMS

**Corresponding constant:**

MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_CONIC_ITEMS

**Description:**

If the output format is not compatible with conic quadratic problems this parameter controls if
the writer ignores the conic parts or produces an error.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

### 23.2.189    MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_ITEMS

**Corresponding constant:**

MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_ITEMS

**Description:**

Controls if the writer ignores incompatible problem items when writing files.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

### 23.2.190 MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_NL_ITEMS

**Corresponding constant:**

MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_NL_ITEMS

**Description:**

Controls if the writer ignores general non-linear terms or produces an error.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

### 23.2.191 MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_PSD_ITEMS

**Corresponding constant:**

MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_PSD_ITEMS

**Description:**

If the output format is not compatible with semidefinite problems this parameter controls if the writer ignores the conic parts or produces an error.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

### 23.2.192 MSK_IPAR_WRITE_INT_CONSTRAINTS

**Corresponding constant:**

MSK_IPAR_WRITE_INT_CONSTRAINTS

**Description:**

Controls whether the constraint section is written to the integer solution file.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_ON

### 23.2.193    MSK_IPAR_WRITE_INT_HEAD

**Corresponding constant:**

   MSK_IPAR_WRITE_INT_HEAD

**Description:**

   Controls whether the header section is written to the integer solution file.

**Possible values:**

   - MSK_OFF Switch the option off.
   - MSK_ON Switch the option on.

**Default value:**

   MSK_ON

### 23.2.194    MSK_IPAR_WRITE_INT_VARIABLES

**Corresponding constant:**

   MSK_IPAR_WRITE_INT_VARIABLES

**Description:**

   Controls whether the variables section is written to the integer solution file.

**Possible values:**

   - MSK_OFF Switch the option off.
   - MSK_ON Switch the option on.

**Default value:**

   MSK_ON

### 23.2.195    MSK_IPAR_WRITE_LP_LINE_WIDTH

**Corresponding constant:**

   MSK_IPAR_WRITE_LP_LINE_WIDTH

**Description:**

   Maximum width of line in an LP file written by MOSEK.

**Possible Values:**

   Any positive number.

**Default value:**

   80

### 23.2.196 MSK_IPAR_WRITE_LP_QUOTED_NAMES

**Corresponding constant:**

MSK_IPAR_WRITE_LP_QUOTED_NAMES

**Description:**

If this option is turned on, then MOSEK will quote invalid LP names when writing an LP file.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_ON

### 23.2.197 MSK_IPAR_WRITE_LP_STRICT_FORMAT

**Corresponding constant:**

MSK_IPAR_WRITE_LP_STRICT_FORMAT

**Description:**

Controls whether LP output files satisfy the LP format strictly.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

### 23.2.198 MSK_IPAR_WRITE_LP_TERMS_PER_LINE

**Corresponding constant:**

MSK_IPAR_WRITE_LP_TERMS_PER_LINE

**Description:**

Maximum number of terms on a single line in an LP file written by MOSEK. 0 means unlimited.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

10

### 23.2.199    MSK_IPAR_WRITE_MPS_INT

**Corresponding constant:**

MSK_IPAR_WRITE_MPS_INT

**Description:**

Controls if marker records are written to the MPS file to indicate whether variables are integer restricted.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_ON

### 23.2.200    MSK_IPAR_WRITE_PRECISION

**Corresponding constant:**

MSK_IPAR_WRITE_PRECISION

**Description:**

Controls the precision with which `double` numbers are printed in the MPS data file. In general it is not worthwhile to use a value higher than 15.

**Possible Values:**

Any number between 0 and +inf.

**Default value:**

8

### 23.2.201    MSK_IPAR_WRITE_SOL_BARVARIABLES

**Corresponding constant:**

MSK_IPAR_WRITE_SOL_BARVARIABLES

**Description:**

Controls whether the symmetric matrix variables section is written to the solution file.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_ON

### 23.2.202    MSK_IPAR_WRITE_SOL_CONSTRAINTS

**Corresponding constant:**

MSK_IPAR_WRITE_SOL_CONSTRAINTS

**Description:**

Controls whether the constraint section is written to the solution file.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_ON

### 23.2.203    MSK_IPAR_WRITE_SOL_HEAD

**Corresponding constant:**

MSK_IPAR_WRITE_SOL_HEAD

**Description:**

Controls whether the header section is written to the solution file.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_ON

### 23.2.204    MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES

**Corresponding constant:**

MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES

**Description:**

Even if the names are invalid MPS names, then they are employed when writing the solution file.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_OFF

### 23.2.205   MSK_IPAR_WRITE_SOL_VARIABLES

**Corresponding constant:**

MSK_IPAR_WRITE_SOL_VARIABLES

**Description:**

Controls whether the variables section is written to the solution file.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_ON

### 23.2.206   MSK_IPAR_WRITE_TASK_INC_SOL

**Corresponding constant:**

MSK_IPAR_WRITE_TASK_INC_SOL

**Description:**

Controls whether the solutions are stored in the task file too.

**Possible values:**

- MSK_OFF Switch the option off.
- MSK_ON Switch the option on.

**Default value:**

MSK_ON

### 23.2.207   MSK_IPAR_WRITE_XML_MODE

**Corresponding constant:**

MSK_IPAR_WRITE_XML_MODE

**Description:**

Controls if linear coefficients should be written by row or column when writing in the XML file format.

**Possible values:**

- MSK_WRITE_XML_MODE_COL Write in column order.
- MSK_WRITE_XML_MODE_ROW Write in row order.

**Default value:**

MSK_WRITE_XML_MODE_ROW

## 23.3 MSKsparame: String parameter types

### 23.3.1 MSK_SPAR_BAS_SOL_FILE_NAME

**Corresponding constant:**
MSK_SPAR_BAS_SOL_FILE_NAME

**Description:**
Name of the `bas` solution file.

**Possible Values:**
Any valid file name.

**Default value:**
""

### 23.3.2 MSK_SPAR_DATA_FILE_NAME

**Corresponding constant:**
MSK_SPAR_DATA_FILE_NAME

**Description:**
Data are read and written to this file.

**Possible Values:**
Any valid file name.

**Default value:**
""

### 23.3.3 MSK_SPAR_DEBUG_FILE_NAME

**Corresponding constant:**
MSK_SPAR_DEBUG_FILE_NAME

**Description:**
MOSEK debug file.

**Possible Values:**
Any valid file name.

**Default value:**
""

### 23.3.4   MSK_SPAR_FEASREPAIR_NAME_PREFIX

**Corresponding constant:**

MSK_SPAR_FEASREPAIR_NAME_PREFIX

**Description:**

Not applicable.

**Possible Values:**

Any valid string.

**Default value:**

"MSK-"

### 23.3.5   MSK_SPAR_FEASREPAIR_NAME_SEPARATOR

**Corresponding constant:**

MSK_SPAR_FEASREPAIR_NAME_SEPARATOR

**Description:**

Not applicable.

**Possible Values:**

Any valid string.

**Default value:**

"-"

### 23.3.6   MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL

**Corresponding constant:**

MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL

**Description:**

The constraint and variable associated with the total weighted sum of violations are each given the name of this parameter postfixed with `CON` and `VAR` respectively.

**Possible Values:**

Any valid string.

**Default value:**

"WSUMVIOL"

### 23.3.7 MSK_SPAR_INT_SOL_FILE_NAME

**Corresponding constant:**

MSK_SPAR_INT_SOL_FILE_NAME

**Description:**

Name of the `int` solution file.

**Possible Values:**

Any valid file name.

**Default value:**

""

### 23.3.8 MSK_SPAR_ITR_SOL_FILE_NAME

**Corresponding constant:**

MSK_SPAR_ITR_SOL_FILE_NAME

**Description:**

Name of the `itr` solution file.

**Possible Values:**

Any valid file name.

**Default value:**

""

### 23.3.9 MSK_SPAR_MIO_DEBUG_STRING

**Corresponding constant:**

MSK_SPAR_MIO_DEBUG_STRING

**Description:**

For internal use only.

**Possible Values:**

Any valid string.

**Default value:**

""

### 23.3.10   MSK_SPAR_PARAM_COMMENT_SIGN

**Corresponding constant:**

MSK_SPAR_PARAM_COMMENT_SIGN

**Description:**

Only the first character in this string is used. It is considered as a start of comment sign in the MOSEK parameter file. Spaces are ignored in the string.

**Possible Values:**

Any valid string.

**Default value:**

"%%"

### 23.3.11   MSK_SPAR_PARAM_READ_FILE_NAME

**Corresponding constant:**

MSK_SPAR_PARAM_READ_FILE_NAME

**Description:**

Modifications to the parameter database is read from this file.

**Possible Values:**

Any valid file name.

**Default value:**

""

### 23.3.12   MSK_SPAR_PARAM_WRITE_FILE_NAME

**Corresponding constant:**

MSK_SPAR_PARAM_WRITE_FILE_NAME

**Description:**

The parameter database is written to this file.

**Possible Values:**

Any valid file name.

**Default value:**

""

### 23.3.13  MSK_SPAR_READ_MPS_BOU_NAME

**Corresponding constant:**

MSK_SPAR_READ_MPS_BOU_NAME

**Description:**

Name of the BOUNDS vector used. An empty name means that the first BOUNDS vector is used.

**Possible Values:**

Any valid MPS name.

**Default value:**

""

### 23.3.14  MSK_SPAR_READ_MPS_OBJ_NAME

**Corresponding constant:**

MSK_SPAR_READ_MPS_OBJ_NAME

**Description:**

Name of the free constraint used as objective function. An empty name means that the first constraint is used as objective function.

**Possible Values:**

Any valid MPS name.

**Default value:**

""

### 23.3.15  MSK_SPAR_READ_MPS_RAN_NAME

**Corresponding constant:**

MSK_SPAR_READ_MPS_RAN_NAME

**Description:**

Name of the RANGE vector used. An empty name means that the first RANGE vector is used.

**Possible Values:**

Any valid MPS name.

**Default value:**

""

### 23.3.16   MSK_SPAR_READ_MPS_RHS_NAME

**Corresponding constant:**

MSK_SPAR_READ_MPS_RHS_NAME

**Description:**

Name of the RHS used. An empty name means that the first RHS vector is used.

**Possible Values:**

Any valid MPS name.

**Default value:**

""

### 23.3.17   MSK_SPAR_SENSITIVITY_FILE_NAME

**Corresponding constant:**

MSK_SPAR_SENSITIVITY_FILE_NAME

**Description:**

Not applicable.

**Possible Values:**

Any valid string.

**Default value:**

""

### 23.3.18   MSK_SPAR_SENSITIVITY_RES_FILE_NAME

**Corresponding constant:**

MSK_SPAR_SENSITIVITY_RES_FILE_NAME

**Description:**

Not applicable.

**Possible Values:**

Any valid string.

**Default value:**

""

### 23.3.19  MSK_SPAR_SOL_FILTER_XC_LOW

**Corresponding constant:**

MSK_SPAR_SOL_FILTER_XC_LOW

**Description:**

A filter used to determine which constraints should be listed in the solution file. A value of "0.5" means that all constraints having `xc[i]>0.5` should be listed, whereas "+0.5" means that all constraints having `xc[i]>=blc[i]+0.5` should be listed. An empty filter means that no filter is applied.

**Possible Values:**

Any valid filter.

**Default value:**

""

### 23.3.20  MSK_SPAR_SOL_FILTER_XC_UPR

**Corresponding constant:**

MSK_SPAR_SOL_FILTER_XC_UPR

**Description:**

A filter used to determine which constraints should be listed in the solution file. A value of "0.5" means that all constraints having `xc[i]<0.5` should be listed, whereas "-0.5" means all constraints having `xc[i]<=buc[i]-0.5` should be listed. An empty filter means that no filter is applied.

**Possible Values:**

Any valid filter.

**Default value:**

""

### 23.3.21  MSK_SPAR_SOL_FILTER_XX_LOW

**Corresponding constant:**

MSK_SPAR_SOL_FILTER_XX_LOW

**Description:**

A filter used to determine which variables should be listed in the solution file. A value of "0.5" means that all constraints having `xx[j]>=0.5` should be listed, whereas "+0.5" means that all constraints having `xx[j]>=blx[j]+0.5` should be listed. An empty filter means no filter is applied.

**Possible Values:**

Any valid filter.

**Default value:**

""

### 23.3.22    MSK_SPAR_SOL_FILTER_XX_UPR

**Corresponding constant:**

MSK_SPAR_SOL_FILTER_XX_UPR

**Description:**

A filter used to determine which variables should be listed in the solution file.   A value of
"0.5" means that all constraints having `xx[j]<0.5` should be printed, whereas "-0.5" means
all constraints having `xx[j]<=bux[j]-0.5` should be listed. An empty filter means no filter is
applied.

**Possible Values:**

Any valid file name.

**Default value:**

""

### 23.3.23    MSK_SPAR_STAT_FILE_NAME

**Corresponding constant:**

MSK_SPAR_STAT_FILE_NAME

**Description:**

Statistics file name.

**Possible Values:**

Any valid file name.

**Default value:**

""

### 23.3.24    MSK_SPAR_STAT_KEY

**Corresponding constant:**

MSK_SPAR_STAT_KEY

**Description:**

Key used when writing the summary file.

**Possible Values:**

Any valid XML string.

**Default value:**

""

### 23.3.25 MSK_SPAR_STAT_NAME

**Corresponding constant:**

MSK_SPAR_STAT_NAME

**Description:**

Name used when writing the statistics file.

**Possible Values:**

Any valid XML string.

**Default value:**

""

### 23.3.26 MSK_SPAR_WRITE_LP_GEN_VAR_NAME

**Corresponding constant:**

MSK_SPAR_WRITE_LP_GEN_VAR_NAME

**Description:**

Sometimes when an LP file is written additional variables must be inserted. They will have the prefix denoted by this parameter.

**Possible Values:**

Any valid string.

**Default value:**

"xmskgen"

# Chapter 24

# Response codes

Response codes ordered by name.

**MSK_RES_ERR_AD_INVALID_CODELIST** (3102)

    The code list data was invalid.

**MSK_RES_ERR_AD_INVALID_OPERAND** (3104)

    The code list data was invalid. An unknown operand was used.

**MSK_RES_ERR_AD_INVALID_OPERATOR** (3103)

    The code list data was invalid. An unknown operator was used.

**MSK_RES_ERR_AD_MISSING_OPERAND** (3105)

    The code list data was invalid. Missing operand for operator.

**MSK_RES_ERR_AD_MISSING_RETURN** (3106)

    The code list data was invalid. Missing return operation in function.

**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_CBF_INVALID_INT_INDEX (7121)

  Invalid INT index.

MSK_RES_ERR_CBF_INVALID_VAR_TYPE (7111)

  Invalid variable type.

MSK_RES_ERR_CBF_NO_VARIABLES (7102)

  No variables are specified.

MSK_RES_ERR_CBF_NO_VERSION_SPECIFIED (7105)

  No version specified.

MSK_RES_ERR_CBF_OBJ_SENSE (7101)

  An invalid objective sense is specified.

MSK_RES_ERR_CBF_PARSE (7100)

  An error occurred while parsing an CBF file.

MSK_RES_ERR_CBF_SYNTAX (7106)

  Invalid syntax.

MSK_RES_ERR_CBF_TOO_FEW_CONSTRAINTS (7118)

  Too few constraints defined.

MSK_RES_ERR_CBF_TOO_FEW_INTS (7119)

  Too few ints are specified.

MSK_RES_ERR_CBF_TOO_FEW_VARIABLES (7117)

  Too few variables defined.

MSK_RES_ERR_CBF_TOO_MANY_CONSTRAINTS (7103)

   Too many constraints specified.

MSK_RES_ERR_CBF_TOO_MANY_INTS (7120)

   Too many ints are specified.

MSK_RES_ERR_CBF_TOO_MANY_VARIABLES (7104)

   Too many variables specified.

MSK_RES_ERR_CBF_UNSUPPORTED (7122)

   Unsupported feature is present.

MSK_RES_ERR_CON_Q_NOT_NSD (1294)

   The quadratic constraint matrix is not negative semidefinite as expected for a constraint with finite lower bound.  This results in a nonconvex problem.  The parameter MSK_DPAR_CHECK_CONVEXITY_REL_TOL can be used to relax the convexity check.

MSK_RES_ERR_CON_Q_NOT_PSD (1293)

   The quadratic constraint matrix is not positive semidefinite as expected for a constraint with finite upper bound.  This results in a nonconvex problem.  The parameter MSK_DPAR_CHECK_CONVEXITY_REL_TOL can be used to relax the convexity check.

MSK_RES_ERR_CONCURRENT_OPTIMIZER (3059)

   An unsupported optimizer was chosen for use with the concurrent optimizer.

MSK_RES_ERR_CONE_INDEX (1300)

   An index of a non-existing cone has been specified.

MSK_RES_ERR_CONE_OVERLAP (1302)

   A new cone which variables overlap with an existing cone has been specified.

MSK_RES_ERR_CONE_OVERLAP_APPEND (1307)

   The cone to be appended has one variable which is already memeber of another cone.

MSK_RES_ERR_CONE_REP_VAR (1303)

   A variable is included multiple times in the cone.

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_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_INTERNAL (3000)

An internal error occurred. Please report this problem.

MSK_RES_ERR_INTERNAL_TEST_FAILED (3500)

An internal unit test function failed.

MSK_RES_ERR_INV_APTRE (1253)

`aptre[j]` is strictly smaller than `aptrb[j]` for some `j`.

MSK_RES_ERR_INV_BK (1255)

Invalid bound key.

MSK_RES_ERR_INV_BKC (1256)

Invalid bound key is specified for a constraint.

MSK_RES_ERR_INV_BKX (1257)

An invalid bound key is specified for a variable.

MSK_RES_ERR_INV_CONE_TYPE (1272)

> Invalid cone type code is encountered.

MSK_RES_ERR_INV_CONE_TYPE_STR (1271)

> Invalid cone type string encountered.

MSK_RES_ERR_INV_CONIC_PROBLEM (1502)

> The conic optimizer can only be applied to problems with linear objective and constraints. Many problems such convex quadratically constrained problems can easily be reformulated to conic problems. See the appropriate MOSEK manual for details.

MSK_RES_ERR_INV_MARKI (2501)

> Invalid value in marki.

MSK_RES_ERR_INV_MARKJ (2502)

> Invalid value in markj.

MSK_RES_ERR_INV_NAME_ITEM (1280)

> An invalid name item code is used.

MSK_RES_ERR_INV_NUMI (2503)

> Invalid numi.

MSK_RES_ERR_INV_NUMJ (2504)

> Invalid numj.

MSK_RES_ERR_INV_OPTIMIZER (1550)

> An invalid optimizer has been chosen for the problem. This means that the simplex or the conic optimizer is chosen to optimize a nonlinear problem.

MSK_RES_ERR_INV_PROBLEM (1500)

> Invalid problem type. Probably a nonconvex problem has been specified.

MSK_RES_ERR_INV_QCON_SUBI (1405)

> Invalid value in qcsubi.

MSK_RES_ERR_INV_QCON_SUBJ (1406)

> Invalid value in qcsubj.

MSK_RES_ERR_INV_QCON_SUBK (1404)

> Invalid value in qcsubk.

MSK_RES_ERR_INV_QCON_VAL (1407)

> Invalid value in qcval.

MSK_RES_ERR_INV_QOBJ_SUBI (1401)

> Invalid value in qosubi.

MSK_RES_ERR_INV_QOBJ_SUBJ (1402)

Invalid value in `qosubj`.

MSK_RES_ERR_INV_QOBJ_VAL (1403)

Invalid value in `qoval`.

MSK_RES_ERR_INV_SK (1270)

Invalid status key code.

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_BRANCH_DIRECTION (3200)

An invalid branching direction is specified.

MSK_RES_ERR_INVALID_BRANCH_PRIORITY (3201)

An invalid branching priority is specified. It should be nonnegative.

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_NETWORK_PROBLEM (1504)

> The problem is not a network problem as expected. The error occurs if a network optimizer is applied to a problem that cannot (easily) be converted to a network problem.

MSK_RES_ERR_INVALID_OBJ_NAME (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_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` (7004)

> Invalid argument k.

`MSK_RES_ERR_LAU_ARG_M` (7002)

> Invalid argument m.

`MSK_RES_ERR_LAU_ARG_N` (7003)

> Invalid argument n.

`MSK_RES_ERR_LAU_ARG_TRANS` (7008)

> Invalid argument trans.

`MSK_RES_ERR_LAU_ARG_TRANSA` (7005)

> Invalid argument transa.

MSK_RES_ERR_LAU_ARG_TRANSB (7006)

> Invalid argument transb.

MSK_RES_ERR_LAU_ARG_UPLO (7007)

> Invalid argument uplo.

MSK_RES_ERR_LAU_SINGULAR_MATRIX (7000)

> A matrix is singular.

MSK_RES_ERR_LAU_UNKNOWN (7001)

> 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 specfified.

**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 $Q$ matrixes is smaller than the number of non-zeros in the current $Q$ matrixes.

**MSK RES ERR MAXNUMVAR** (1241)

The maximum number of variables specified is smaller than the number of variables in the task.

**MSK RES ERR MBT INCOMPATIBLE** (2550)

The MBT file is incompatible with this platform. This results from reading a file on a 32 bit platform generated on a 64 bit platform.

**MSK RES ERR MBT INVALID** (2551)

The MBT file is invalid.

**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 file or a token server. See the MOSEK installation manual for details.

MSK_RES_ERR_MIXED_PROBLEM (1501)

   The problem contains both conic and nonlinear constraints.

MSK_RES_ERR_MPS_CONE_OVERLAP (1118)

   A variable is specified to be a member of several cones.

MSK_RES_ERR_MPS_CONE_REPEAT (1119)

   A variable is repeated within the CSECTION.

MSK_RES_ERR_MPS_CONE_TYPE (1117)

   Invalid cone type specified in a CSECTION.

MSK_RES_ERR_MPS_DUPLICATE_Q_ELEMENT (1121)

   Duplicate elements is specfied in a $Q$ matrix.

MSK_RES_ERR_MPS_FILE (1100)

   An error occurred while reading an MPS file.

MSK_RES_ERR_MPS_INV_BOUND_KEY (1108)

   An invalid bound key occurred in an MPS file.

MSK_RES_ERR_MPS_INV_CON_KEY (1107)

   An invalid constraint key occurred in an MPS file.

MSK_RES_ERR_MPS_INV_FIELD (1101)

   A field in the MPS file is invalid. Probably it is too wide.

MSK_RES_ERR_MPS_INV_MARKER (1102)

   An invalid marker has been specified in the MPS file.

MSK_RES_ERR_MPS_INV_SEC_NAME (1109)

   An invalid section name occurred in an MPS file.

MSK_RES_ERR_MPS_INV_SEC_ORDER (1115)

   The sections in the MPS data file are not in the correct order.

MSK_RES_ERR_MPS_INVALID_OBJ_NAME (1128)

> An invalid objective name is specified.

MSK_RES_ERR_MPS_INVALID_OBJSENSE (1122)

> An invalid objective sense is specified.

MSK_RES_ERR_MPS_MUL_CON_NAME (1112)

> A constraint name was specified 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 QSECTIONs 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.

MSK_RES_ERR_NEWER_DLL (1036)

>   The dynamic link library is newer than the specified version.

MSK_RES_ERR_NO_BARS_FOR_SOLUTION (3916)

>   There is no $\bar{s}$ available for the solution specified. In particular note there are no $\bar{s}$ defined for the basic and integer solutions.

MSK_RES_ERR_NO_BARX_FOR_SOLUTION (3915)

>   There is no $\bar{X}$ available for the solution specified. In particular note there are no $\bar{X}$ defined for the basic and integer solutions.

MSK_RES_ERR_NO_BASIS_SOL (1600)

> No basic solution is defined.

MSK_RES_ERR_NO_DUAL_FOR_ITG_SOL (2950)

> No dual information is available for the integer solution.

MSK_RES_ERR_NO_DUAL_INFEAS_CER (2001)

> A certificate of infeasibility is not available.

MSK_RES_ERR_NO_DUAL_INFO_FOR_ITG_SOL (3300)

> Dual information is not available for the integer solution.

MSK_RES_ERR_NO_INIT_ENV (1063)

> env is not initialized.

MSK_RES_ERR_NO_OPTIMIZER_VAR_TYPE (1552)

> No optimizer is available for this class of optimization problems.

MSK_RES_ERR_NO_PRIMAL_INFEAS_CER (2000)

> A certificate of primal infeasibility is not available.

MSK_RES_ERR_NO_SNX_FOR_BAS_SOL (2953)

> $s_n^x$ is not available for the basis solution.

MSK_RES_ERR_NO_SOLUTION_IN_CALLBACK (2500)

> The required solution is not available.

MSK_RES_ERR_NON_UNIQUE_ARRAY (5000)

> An array does not contain unique elements.

MSK_RES_ERR_NONCONVEX (1291)

> The optimization problem is nonconvex.

MSK_RES_ERR_NONLINEAR_EQUALITY (1290)

> The model contains a nonlinear equality which defines a nonconvex set.

MSK_RES_ERR_NONLINEAR_FUNCTIONS_NOT_ALLOWED (1428)

> An operation that is invalid for problems with nonlinear functions defined has been attempted.

MSK_RES_ERR_NONLINEAR_RANGED (1292)

> The model contains a nonlinear ranged constraint which by definition defines a nonconvex set.

MSK_RES_ERR_NR_ARGUMENTS (1199)

> Incorrect number of function arguments.

MSK_RES_ERR_NULL_ENV (1060)

> env is a NULL pointer.

**MSK_RES_ERR_NULL_POINTER (1065)**

> An argument to a function is unexpectedly a NULL pointer.

**MSK_RES_ERR_NULL_TASK (1061)**

> `task` is a NULL pointer.

**MSK_RES_ERR_NUMCONLIM (1250)**

> Maximum number of constraints limit is exceeded.

**MSK_RES_ERR_NUMVARLIM (1251)**

> Maximum number of variables limit is exceeded.

**MSK_RES_ERR_OBJ_Q_NOT_NSD (1296)**

> The quadratic coefficient matrix in the objective is not negative semidefinite as expected for a maximization problem. The parameter MSK_DPAR_CHECK_CONVEXITY_REL_TOL can be used to relax the convexity check.

**MSK_RES_ERR_OBJ_Q_NOT_PSD (1295)**

> The quadratic coefficient matrix in the objective is not positive semidefinite as expected for a minimization problem. The parameter MSK_DPAR_CHECK_CONVEXITY_REL_TOL can be used to relax the convexity check.

**MSK_RES_ERR_OBJECTIVE_RANGE (1260)**

> Empty objective range.

**MSK_RES_ERR_OLDER_DLL (1035)**

> The dynamic link library is older than the specified version.

**MSK_RES_ERR_OPEN_DL (1030)**

> A dynamic link library could not be opened.

**MSK_RES_ERR_OPF_FORMAT (1168)**

> Syntax error in an OPF file

**MSK_RES_ERR_OPF_NEW_VARIABLE (1169)**

> Introducing new variables is now allowed. When a `[variables]` section is present, it is not allowed to introduce new variables later in the problem.

**MSK_RES_ERR_OPF_PREMATURE_EOF (1172)**

> Premature end of file in an OPF file.

**MSK_RES_ERR_OPTIMIZER_LICENSE (1013)**

> The optimizer required is not licensed.

**MSK_RES_ERR_ORD_INVALID (1131)**

> Invalid content in branch ordering file.

MSK_RES_ERR_ORD_INVALID_BRANCH_DIR (1130)

  An invalid branch direction key is specified.

MSK_RES_ERR_OVERFLOW (1590)

  A computation produced an overflow i.e. a very large number.

MSK_RES_ERR_PARAM_INDEX (1210)

  Parameter index is out of range.

MSK_RES_ERR_PARAM_IS_TOO_LARGE (1215)

  The parameter value is too large.

MSK_RES_ERR_PARAM_IS_TOO_SMALL (1216)

  The parameter value is too small.

MSK_RES_ERR_PARAM_NAME (1205)

  The parameter name is not correct.

MSK_RES_ERR_PARAM_NAME_DOU (1206)

  The parameter name is not correct for a double parameter.

MSK_RES_ERR_PARAM_NAME_INT (1207)

  The parameter name is not correct for an integer parameter.

MSK_RES_ERR_PARAM_NAME_STR (1208)

  The parameter name is not correct for a string parameter.

MSK_RES_ERR_PARAM_TYPE (1218)

  The parameter type is invalid.

MSK_RES_ERR_PARAM_VALUE_STR (1217)

  The parameter value string is incorrect.

MSK_RES_ERR_PLATFORM_NOT_LICENSED (1019)

  A requested license feature is not available for the required platform.

MSK_RES_ERR_POSTSOLVE (1580)

  An error occurred during the postsolve. Please contact MOSEK support.

MSK_RES_ERR_PRO_ITEM (1281)

  An invalid problem is used.

MSK_RES_ERR_PROB_LICENSE (1006)

  The software is not licensed to solve the problem.

MSK_RES_ERR_QCON_SUBI_TOO_LARGE (1409)

  Invalid value in `qcsubi`.

MSK_RES_ERR_QCON_SUBI_TOO_SMALL (1408)

> Invalid value in `qcsubi`.

MSK_RES_ERR_QCON_UPPER_TRIANGLE (1417)

> An element in the upper triangle of a $Q^k$ is specified. Only elements in the lower triangle should be specified.

MSK_RES_ERR_QOBJ_UPPER_TRIANGLE (1415)

> An element in the upper triangle of $Q^o$ is specified. Only elements in the lower triangle should be specified.

MSK_RES_ERR_READ_FORMAT (1090)

> The specified format cannot be read.

MSK_RES_ERR_READ_LP_MISSING_END_TAG (1159)

> Syntax error in LP file. Possibly missing End tag.

MSK_RES_ERR_READ_LP_NONEXISTING_NAME (1162)

> A variable never occurred in objective or constraints.

MSK_RES_ERR_REMOVE_CONE_VARIABLE (1310)

> A variable cannot be removed because it will make a cone invalid.

MSK_RES_ERR_REPAIR_INVALID_PROBLEM (1710)

> The feasibility repair does not support the specified problem type.

MSK_RES_ERR_REPAIR_OPTIMIZATION_FAILED (1711)

> Computation the optimal relaxation failed. The cause may have been numerical problems.

MSK_RES_ERR_SEN_BOUND_INVALID_LO (3054)

> Analysis of lower bound requested for an index, where no lower bound exists.

MSK_RES_ERR_SEN_BOUND_INVALID_UP (3053)

> Analysis of upper bound requested for an index, where no upper bound exists.

MSK_RES_ERR_SEN_FORMAT (3050)

> Syntax error in sensitivity analysis file.

MSK_RES_ERR_SEN_INDEX_INVALID (3055)

> Invalid range given in the sensitivity file.

MSK_RES_ERR_SEN_INDEX_RANGE (3052)

> Index out of range in the sensitivity analysis file.

MSK_RES_ERR_SEN_INVALID_REGEXP (3056)

> Syntax error in regexp or regexp longer than 1024.

MSK_RES_ERR_SEN_NUMERICAL (3058)

  Numerical difficulties encountered performing the sensitivity analysis.

MSK_RES_ERR_SEN_SOLUTION_STATUS (3057)

  No optimal solution found to the original problem given for sensitivity analysis.

MSK_RES_ERR_SEN_UNDEF_NAME (3051)

  An undefined name was encountered in the sensitivity analysis file.

MSK_RES_ERR_SEN_UNHANDLED_PROBLEM_TYPE (3080)

  Sensitivity analysis cannot be performed for the spcified problem.  Sensitivity analysis is only
  possible for linear problems.

MSK_RES_ERR_SIZE_LICENSE (1005)

  The problem is bigger than the license.

MSK_RES_ERR_SIZE_LICENSE_CON (1010)

  The problem has too many constraints to be solved with the available license.

MSK_RES_ERR_SIZE_LICENSE_INTVAR (1012)

  The problem contains too many integer variables to be solved with the available license.

MSK_RES_ERR_SIZE_LICENSE_NUMCORES (3900)

  The computer contains more cpu cores than the license allows for.

MSK_RES_ERR_SIZE_LICENSE_VAR (1011)

  The problem has too many variables to be solved with the available license.

MSK_RES_ERR_SOL_FILE_INVALID_NUMBER (1350)

  An invalid number is specified in a solution file.

MSK_RES_ERR_SOLITEM (1237)

  The solution item number `solitem` is invalid.  Please note that MSK_SOL_ITEM_SNX is invalid for
  the basic solution.

MSK_RES_ERR_SOLVER_PROBTYPE (1259)

  Problem type does not match the chosen optimizer.

MSK_RES_ERR_SPACE (1051)

  Out of space.

MSK_RES_ERR_SPACE_LEAKING (1080)

  MOSEK is leaking memory. This can be due to either an incorrect use of MOSEK or a bug.

MSK_RES_ERR_SPACE_NO_INFO (1081)

  No available information about the space usage.

**MSK_RES_ERR_SYM_MAT_DUPLICATE (3944)**

A value in a symmetric matric as been specified more than once.

**MSK_RES_ERR_SYM_MAT_INVALID_COL_INDEX (3941)**

A column index specified for sparse symmetric maxtrix is invalid.

**MSK_RES_ERR_SYM_MAT_INVALID_ROW_INDEX (3940)**

A row index specified for sparse symmetric maxtrix is invalid.

**MSK_RES_ERR_SYM_MAT_INVALID_VALUE (3943)**

The numerical value specified in a sparse symmetric matrix is not a value floating value.

**MSK_RES_ERR_SYM_MAT_NOT_LOWER_TRINGULAR (3942)**

Only the lower triangular part of sparse symmetric matrix should be specified.

**MSK_RES_ERR_TASK_INCOMPATIBLE (2560)**

The Task file is incompatible with this platform. This results from reading a file on a 32 bit platform generated on a 64 bit platform.

**MSK_RES_ERR_TASK_INVALID (2561)**

The Task file is invalid.

**MSK_RES_ERR_THREAD_COND_INIT (1049)**

Could not initialize a condition.

**MSK_RES_ERR_THREAD_CREATE (1048)**

Could not create a thread. This error may occur if a large number of environments are created and not deleted again. In any case it is a good practice to minimize the number of environments created.

**MSK_RES_ERR_THREAD_MUTEX_INIT (1045)**

Could not initialize a mutex.

**MSK_RES_ERR_THREAD_MUTEX_LOCK (1046)**

Could not lock a mutex.

**MSK_RES_ERR_THREAD_MUTEX_UNLOCK (1047)**

Could not unlock a mutex.

**MSK_RES_ERR_TOCONIC_CONVERSION_FAIL (7200)**

A constraint could not be converted in conic form.

**MSK_RES_ERR_TOO_MANY_CONCURRENT_TASKS (3090)**

Too many concurrent tasks specified.

**MSK_RES_ERR_TOO_SMALL_MAX_NUM_NZ (1245)**

The maximum number of non-zeros specified is too small.

MSK_RES_ERR_TOO_SMALL_MAXNUMANZ (1252)

The maximum number of non-zeros specified for $A$ is smaller than the number of non-zeros in the current $A$.

MSK_RES_ERR_UNB_STEP_SIZE (3100)

A step size in an optimizer was unexpectedly unbounded. For instance, if the step-size becomes unbounded in phase 1 of the simplex algorithm then an error occurs. Normally this will happen only if the problem is badly formulated. Please contact MOSEK support if this error occurs.

MSK_RES_ERR_UNDEF_SOLUTION (1265)

MOSEK has the following solution types:

- an interior-point solution,
- an basic solution,
- and an integer solution.

Each optimizer may set one or more of these solutions; e.g by default a successful optimization with the interior-point optimizer defines the interior-point solution, and, for linear problems, also the basic solution. This error occurs when asking for a solution or for information about a solution that is not defined.

MSK_RES_ERR_UNDEFINED_OBJECTIVE_SENSE (1446)

The objective sense has not been specified before the optimization.

MSK_RES_ERR_UNHANDLED_SOLUTION_STATUS (6010)

Unhandled solution status.

MSK_RES_ERR_UNKNOWN (1050)

Unknown error.

MSK_RES_ERR_UPPER_BOUND_IS_A_NAN (1391)

The upper bound specified is not a number (nan).

MSK_RES_ERR_UPPER_TRIANGLE (6020)

An element in the upper triangle of a lower triangular matrix is specified.

MSK_RES_ERR_USER_FUNC_RET (1430)

An user function reported an error.

MSK_RES_ERR_USER_FUNC_RET_DATA (1431)

An user function returned invalid data.

MSK_RES_ERR_USER_NLO_EVAL (1433)

The user-defined nonlinear function reported an error.

MSK_RES_ERR_USER_NLO_EVAL_HESSUBI (1440)

The user-defined nonlinear function reported an invalid subscript in the Hessian.

**MSK_RES_ERR_USER_NLO_EVAL_HESSUBJ (1441)**

   The user-defined nonlinear function reported an invalid subscript in the Hessian.

**MSK_RES_ERR_USER_NLO_FUNC (1432)**

   The user-defined nonlinear function reported an error.

**MSK_RES_ERR_WHICHITEM_NOT_ALLOWED (1238)**

   `whichitem` is unacceptable.

**MSK_RES_ERR_WHICHSOL (1236)**

   The solution defined by compwhichsol does not exists.

**MSK_RES_ERR_WRITE_LP_FORMAT (1158)**

   Problem cannot be written as an LP file.

**MSK_RES_ERR_WRITE_LP_NON_UNIQUE_NAME (1161)**

   An auto-generated name is not unique.

**MSK_RES_ERR_WRITE_MPS_INVALID_NAME (1153)**

   An invalid name is created while writing an MPS file. Usually this will make the MPS file unreadable.

**MSK_RES_ERR_WRITE_OPF_INVALID_VAR_NAME (1156)**

   Empty variable names cannot be written to OPF files.

**MSK_RES_ERR_WRITING_FILE (1166)**

   An error occurred while writing file

**MSK_RES_ERR_XML_INVALID_PROBLEM_TYPE (3600)**

   The problem type is not supported by the XML format.

**MSK_RES_ERR_Y_IS_UNDEFINED (1449)**

   The solution item $y$ is undefined.

**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
   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.

MSK_RES_WRN_ANA_ALMOST_INT_BOUNDS (904)

> This warning is issued by the problem analyzer if a constraint is bound nearly integral.

MSK_RES_WRN_ANA_C_ZERO (901)

> This warning is issued by the problem analyzer, if the coefficients in the linear part of the objective are all zero.

MSK_RES_WRN_ANA_CLOSE_BOUNDS (903)

> This warning is issued by problem analyzer, if ranged constraints or variables with very close upper and lower bounds are detected. One should consider treating such constraints as equalities and such variables as constants.

MSK_RES_WRN_ANA_EMPTY_COLS (902)

> This warning is issued by the problem analyzer, if columns, in which all coefficients are zero, are found.

MSK_RES_WRN_ANA_LARGE_BOUNDS (900)

> This warning is issued by the problem analyzer, if one or more constraint or variable bounds are very large. One should consider omitting these bounds entirely by setting them to +inf or -inf.

MSK_RES_WRN_CONSTRUCT_INVALID_SOL_ITG (807)

> The intial value for one or more of the integer variables is not feasible.

MSK_RES_WRN_CONSTRUCT_NO_SOL_ITG (810)

> The construct solution requires an integer solution.

MSK_RES_WRN_CONSTRUCT_SOLUTION_INFEAS (805)

> After fixing the integer variables at the suggested values then the problem is infeasible.

MSK_RES_WRN_DROPPED_NZ_QOBJ (201)

> One or more non-zero elements were dropped in the Q matrix in the objective.

MSK_RES_WRN_DUPLICATE_BARVARIABLE_NAMES (852)

> Two barvariable names are identical.

MSK_RES_WRN_DUPLICATE_CONE_NAMES (853)

> Two cone names are identical.

MSK_RES_WRN_DUPLICATE_CONSTRAINT_NAMES (850)

> Two constraint names are identical.

MSK_RES_WRN_DUPLICATE_VARIABLE_NAMES (851)

> Two variable names are identical.

**MSK_RES_WRN_ELIMINATOR_SPACE** (801)

   The eliminator is skipped at least once due to lack of space.

**MSK_RES_WRN_EMPTY_NAME** (502)

   A variable or constraint name is empty. The output file may be invalid.

**MSK_RES_WRN_IGNORE_INTEGER** (250)

   Ignored integer constraints.

**MSK_RES_WRN_INCOMPLETE_LINEAR_DEPENDENCY_CHECK** (800)

   The linear dependency check(s) is not completed. Normally this is not an important warning unless the optimization problem has been formulated with linear dependencies which is bad practice.

**MSK_RES_WRN_LARGE_AIJ** (62)

   A numerically large value is specified for an $a_{i,j}$ element in $A$. The parameter **MSK_DPAR_DATA_TOL_AIJ_LARGE** controls when an $a_{i,j}$ is considered large.

**MSK_RES_WRN_LARGE_BOUND** (51)

   A numerically large bound value is specified.

**MSK_RES_WRN_LARGE_CJ** (57)

   A numerically large value is specified for one $c_j$.

**MSK_RES_WRN_LARGE_CON_FX** (54)

   An equality constraint is fixed to a numerically large value. This can cause numerical problems.

**MSK_RES_WRN_LARGE_LO_BOUND** (52)

   A numerically large lower bound value is specified.

**MSK_RES_WRN_LARGE_UP_BOUND** (53)

   A numerically large upper bound value is specified.

**MSK_RES_WRN_LICENSE_EXPIRE** (500)

   The license expires.

**MSK_RES_WRN_LICENSE_FEATURE_EXPIRE** (505)

   The license expires.

**MSK_RES_WRN_LICENSE_SERVER** (501)

   The license server is not responding.

**MSK_RES_WRN_LP_DROP_VARIABLE** (85)

   Ignored a variable because the variable was not previously defined. Usually this implies that a variable appears in the bound section but not in the objective or the constraints.

**MSK_RES_WRN_LP_OLD_QUAD_FORMAT (80)**

Missing '/2' after quadratic expressions in bound or objective.

**MSK_RES_WRN_MIO_INFEASIBLE_FINAL (270)**

The final mixed-integer problem with all the integer variables fixed at their optimal values is infeasible.

**MSK_RES_WRN_MPS_SPLIT_BOU_VECTOR (72)**

A BOUNDS vector is split into several nonadjacent parts in an MPS file.

**MSK_RES_WRN_MPS_SPLIT_RAN_VECTOR (71)**

A RANGE vector is split into several nonadjacent parts in an MPS file.

**MSK_RES_WRN_MPS_SPLIT_RHS_VECTOR (70)**

An RHS vector is split into several nonadjacent parts in an MPS file.

**MSK_RES_WRN_NAME_MAX_LEN (65)**

A name is longer than the buffer that is supposed to hold it.

**MSK_RES_WRN_NO_DUALIZER (950)**

No automatic dualizer is available for the specified problem. The primal problem is solved.

**MSK_RES_WRN_NO_GLOBAL_OPTIMIZER (251)**

No global optimizer is available.

**MSK_RES_WRN_NO_NONLINEAR_FUNCTION_WRITE (450)**

The problem contains a general nonlinear function in either the objective or the constraints. Such a nonlinear function cannot be written to a disk file. Note that quadratic terms when inputted explicitly can be written to disk.

**MSK_RES_WRN_NZ_IN_UPR_TRI (200)**

Non-zero elements specified in the upper triangle of a matrix were ignored.

**MSK_RES_WRN_OPEN_PARAM_FILE (50)**

The parameter file could not be opened.

**MSK_RES_WRN_PARAM_IGNORED_CMIO (516)**

A parameter was ignored by the conic mixed integer optimizer.

**MSK_RES_WRN_PARAM_NAME_DOU (510)**

The parameter name is not recognized as a double parameter.

**MSK_RES_WRN_PARAM_NAME_INT (511)**

The parameter name is not recognized as a integer parameter.

**MSK_RES_WRN_PARAM_NAME_STR (512)**

The parameter name is not recognized as a string parameter.

MSK_RES_WRN_PARAM_STR_VALUE (515)

> The string is not recognized as a symbolic value for the parameter.

MSK_RES_WRN_PRESOLVE_OUTOFSPACE (802)

> The presolve is incomplete due to lack of space.

MSK_RES_WRN_QUAD_CONES_WITH_ROOT_FIXED_AT_ZERO (930)

> For at least one quadratic cone the root is fixed at (nearly) zero. This may cause problems such as a very large dual solution. Therefore, it is recommended to remove such cones before optimizing the problems, or to fix all the variables in the cone to 0.

MSK_RES_WRN_RQUAD_CONES_WITH_ROOT_FIXED_AT_ZERO (931)

> For at least one rotated quadratic cone at least one of the root variables are fixed at (nearly) zero. This may cause problems such as a very large dual solution. Therefore, it is recommended to remove such cones before optimizing the problems, or to fix all the variables in the cone to 0.

MSK_RES_WRN_SOL_FILE_IGNORED_CON (351)

> One or more lines in the constraint section were ignored when reading a solution file.

MSK_RES_WRN_SOL_FILE_IGNORED_VAR (352)

> One or more lines in the variable section were ignored when reading a solution file.

MSK_RES_WRN_SOL_FILTER (300)

> Invalid solution filter is specified.

MSK_RES_WRN_SPAR_MAX_LEN (66)

> A value for a string parameter is longer than the buffer that is supposed to hold it.

MSK_RES_WRN_TOO_FEW_BASIS_VARS (400)

> An incomplete basis has been specified. Too few basis variables are specified.

MSK_RES_WRN_TOO_MANY_BASIS_VARS (405)

> A basis with too many variables has been specified.

MSK_RES_WRN_TOO_MANY_THREADS_CONCURRENT (750)

> The concurrent optimizer employs more threads than available. This will lead to poor performance.

MSK_RES_WRN_UNDEF_SOL_FILE_NAME (350)

> Undefined name occurred in a solution.

MSK_RES_WRN_USING_GENERIC_NAMES (503)

> Generic names are used because a name is not valid. For instance when writing an LP file the names must not contain blanks or start with a digit.

MSK_RES_WRN_WRITE_CHANGED_NAMES (803)

Some names were changed because they were invalid for the output file format.

**MSK_RES_WRN_WRITE_DISCARDED_CFIX (804)**

The fixed objective term could not be converted to a variable and was discarded in the output file.

**MSK_RES_WRN_ZERO_AIJ (63)**

One or more zero elements are specified in A.

**MSK_RES_WRN_ZEROS_IN_SPARSE_COL (710)**

One or more (near) zero elements are specified in a sparse column of a matrix. It is redundant to specify zero elements. Hence, it may indicate an error.

**MSK_RES_WRN_ZEROS_IN_SPARSE_ROW (705)**

One or more (near) zero elements are specified in a sparse row of a matrix. It is redundant to specify zero elements. Hence it may indicate an error.

# Chapter 25

# API constants

## 25.1 Constraint or variable access modes

**MSK_ACC_VAR**

>   Access data by columns (variable oriented)

**MSK_ACC_CON**

>   Access data by rows (constraint oriented)

## 25.2 Basis identification

**MSK_BI_NEVER**

>   Never do basis identification.

**MSK_BI_ALWAYS**

>   Basis identification is always performed even if the interior-point optimizer terminates abnormally.

**MSK_BI_NO_ERROR**

>   Basis identification is performed if the interior-point optimizer terminates without an error.

**MSK_BI_IF_FEASIBLE**

>   Basis identification is not performed if the interior-point optimizer terminates with a problem status saying that the problem is primal or dual infeasible.

**MSK_BI_RESERVERED**

>   Not currently in use.

## 25.3   Bound keys

`MSK_BK_LO`

> The constraint or variable has a finite lower bound and an infinite upper bound.

`MSK_BK_UP`

> The constraint or variable has an infinite lower bound and an finite upper bound.

`MSK_BK_FX`

> The constraint or variable is fixed.

`MSK_BK_FR`

> The constraint or variable is free.

`MSK_BK_RA`

> The constraint or variable is ranged.

## 25.4   Specifies the branching direction.

`MSK_BRANCH_DIR_FREE`

> The mixed-integer optimizer decides which branch to choose.

`MSK_BRANCH_DIR_UP`

> The mixed-integer optimizer always chooses the up branch first.

`MSK_BRANCH_DIR_DOWN`

> The mixed-integer optimizer always chooses the down branch first.

## 25.5   Progress call-back codes

`MSK_CALLBACK_BEGIN_BI`

> The basis identification procedure has been started.

`MSK_CALLBACK_BEGIN_CONCURRENT`

> Concurrent optimizer is started.

`MSK_CALLBACK_BEGIN_CONIC`

> The call-back function is called when the conic optimizer is started.

`MSK_CALLBACK_BEGIN_DUAL_BI`

> The call-back function is called from within the basis identification procedure when the dual phase is started.

**MSK_CALLBACK_BEGIN_DUAL_SENSITIVITY**

> Dual sensitivity analysis is started.

**MSK_CALLBACK_BEGIN_DUAL_SETUP_BI**

> The call-back function is called when the dual BI phase is started.

**MSK_CALLBACK_BEGIN_DUAL_SIMPLEX**

> The call-back function is called when the dual simplex optimizer started.

**MSK_CALLBACK_BEGIN_DUAL_SIMPLEX_BI**

> The call-back function is called from within the basis identification procedure when the dual simplex clean-up phase is started.

**MSK_CALLBACK_BEGIN_FULL_CONVEXITY_CHECK**

> Begin full convexity check.

**MSK_CALLBACK_BEGIN_INFEAS_ANA**

> The call-back function is called when the infeasibility analyzer is started.

**MSK_CALLBACK_BEGIN_INTPNT**

> The call-back function is called when the interior-point optimizer is started.

**MSK_CALLBACK_BEGIN_LICENSE_WAIT**

> Begin waiting for license.

**MSK_CALLBACK_BEGIN_MIO**

> The call-back function is called when the mixed-integer optimizer is started.

**MSK_CALLBACK_BEGIN_NETWORK_DUAL_SIMPLEX**

> The call-back function is called when the dual network simplex optimizer is started.

**MSK_CALLBACK_BEGIN_NETWORK_PRIMAL_SIMPLEX**

> The call-back function is called when the primal network simplex optimizer is started.

**MSK_CALLBACK_BEGIN_NETWORK_SIMPLEX**

> The call-back function is called when the simplex network optimizer is started.

**MSK_CALLBACK_BEGIN_NONCONVEX**

> The call-back function is called when the nonconvex optimizer is started.

**MSK_CALLBACK_BEGIN_OPTIMIZER**

> The call-back function is called when the optimizer is started.

**MSK_CALLBACK_BEGIN_PRESOLVE**

> The call-back function is called when the presolve is started.

**MSK_CALLBACK_BEGIN_PRIMAL_BI**

> The call-back function is called from within the basis identification procedure when the primal phase is started.

**MSK_CALLBACK_BEGIN_PRIMAL_DUAL_SIMPLEX**

> The call-back function is called when the primal-dual simplex optimizer is started.

**MSK_CALLBACK_BEGIN_PRIMAL_DUAL_SIMPLEX_BI**

> The call-back function is called from within the basis identification procedure when the primal-dual simplex clean-up phase is started.

**MSK_CALLBACK_BEGIN_PRIMAL_REPAIR**

> Begin primal feasibility repair.

**MSK_CALLBACK_BEGIN_PRIMAL_SENSITIVITY**

> Primal sensitivity analysis is started.

**MSK_CALLBACK_BEGIN_PRIMAL_SETUP_BI**

> The call-back function is called when the primal BI setup is started.

**MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX**

> The call-back function is called when the primal simplex optimizer is started.

**MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX_BI**

> The call-back function is called from within the basis identification procedure when the primal simplex clean-up phase is started.

**MSK_CALLBACK_BEGIN_QCQO_REFORMULATE**

> Begin QCQO reformulation.

**MSK_CALLBACK_BEGIN_READ**

> MOSEK has started reading a problem file.

**MSK_CALLBACK_BEGIN_SIMPLEX**

> The call-back function is called when the simplex optimizer is started.

**MSK_CALLBACK_BEGIN_SIMPLEX_BI**

> The call-back function is called from within the basis identification procedure when the simplex clean-up phase is started.

**MSK_CALLBACK_BEGIN_SIMPLEX_NETWORK_DETECT**

> The call-back function is called when the network detection procedure is started.

**MSK_CALLBACK_BEGIN_WRITE**

> MOSEK has started writing a problem file.

**MSK_CALLBACK_CONIC**

> The call-back function is called from within the conic optimizer after the information database has been updated.

**MSK_CALLBACK_DUAL_SIMPLEX**

> The call-back function is called from within the dual simplex optimizer.

**MSK_CALLBACK_END_BI**

> The call-back function is called when the basis identification procedure is terminated.

**MSK_CALLBACK_END_CONCURRENT**

> Concurrent optimizer is terminated.

**MSK_CALLBACK_END_CONIC**

> The call-back function is called when the conic optimizer is terminated.

**MSK_CALLBACK_END_DUAL_BI**

> The call-back function is called from within the basis identification procedure when the dual phase is terminated.

**MSK_CALLBACK_END_DUAL_SENSITIVITY**

> Dual sensitivity analysis is terminated.

**MSK_CALLBACK_END_DUAL_SETUP_BI**

> The call-back function is called when the dual BI phase is terminated.

**MSK_CALLBACK_END_DUAL_SIMPLEX**

> The call-back function is called when the dual simplex optimizer is terminated.

**MSK_CALLBACK_END_DUAL_SIMPLEX_BI**

> The call-back function is called from within the basis identification procedure when the dual clean-up phase is terminated.

**MSK_CALLBACK_END_FULL_CONVEXITY_CHECK**

> End full convexity check.

**MSK_CALLBACK_END_INFEAS_ANA**

> The call-back function is called when the infeasibility analyzer is terminated.

**MSK_CALLBACK_END_INTPNT**

> The call-back function is called when the interior-point optimizer is terminated.

**MSK_CALLBACK_END_LICENSE_WAIT**

> End waiting for license.

**MSK_CALLBACK_END_MIO**

> The call-back function is called when the mixed-integer optimizer is terminated.

MSK_CALLBACK_END_NETWORK_DUAL_SIMPLEX

    The call-back function is called when the dual network simplex optimizer is terminated.

MSK_CALLBACK_END_NETWORK_PRIMAL_SIMPLEX

    The call-back function is called when the primal network simplex optimizer is terminated.

MSK_CALLBACK_END_NETWORK_SIMPLEX

    The call-back function is called when the simplex network optimizer is terminated.

MSK_CALLBACK_END_NONCONVEX

    The call-back function is called when the nonconvex optimizer is terminated.

MSK_CALLBACK_END_OPTIMIZER

    The call-back function is called when the optimizer is terminated.

MSK_CALLBACK_END_PRESOLVE

    The call-back function is called when the presolve is completed.

MSK_CALLBACK_END_PRIMAL_BI

    The call-back function is called from within the basis identification procedure when the primal
    phase is terminated.

MSK_CALLBACK_END_PRIMAL_DUAL_SIMPLEX

    The call-back function is called when the primal-dual simplex optimizer is terminated.

MSK_CALLBACK_END_PRIMAL_DUAL_SIMPLEX_BI

    The call-back function is called from within the basis identification procedure when the primal-
    dual clean-up phase is terminated.

MSK_CALLBACK_END_PRIMAL_REPAIR

    End primal feasibility repair.

MSK_CALLBACK_END_PRIMAL_SENSITIVITY

    Primal sensitivity analysis is terminated.

MSK_CALLBACK_END_PRIMAL_SETUP_BI

    The call-back function is called when the primal BI setup is terminated.

MSK_CALLBACK_END_PRIMAL_SIMPLEX

    The call-back function is called when the primal simplex optimizer is terminated.

MSK_CALLBACK_END_PRIMAL_SIMPLEX_BI

    The call-back function is called from within the basis identification procedure when the primal
    clean-up phase is terminated.

MSK_CALLBACK_END_QCQO_REFORMULATE

    End QCQO reformulation.

**MSK_CALLBACK_END_READ**

MOSEK has finished reading a problem file.

**MSK_CALLBACK_END_SIMPLEX**

The call-back function is called when the simplex optimizer is terminated.

**MSK_CALLBACK_END_SIMPLEX_BI**

The call-back function is called from within the basis identification procedure when the simplex clean-up phase is terminated.

**MSK_CALLBACK_END_SIMPLEX_NETWORK_DETECT**

The call-back function is called when the network detection procedure is terminated.

**MSK_CALLBACK_END_WRITE**

MOSEK has finished writing a problem file.

**MSK_CALLBACK_IM_BI**

The call-back function is called from within the basis identification procedure at an intermediate point.

**MSK_CALLBACK_IM_CONIC**

The call-back function is called at an intermediate stage within the conic optimizer where the information database has not been updated.

**MSK_CALLBACK_IM_DUAL_BI**

The call-back function is called from within the basis identification procedure at an intermediate point in the dual phase.

**MSK_CALLBACK_IM_DUAL_SENSIVITY**

The call-back function is called at an intermediate stage of the dual sensitivity analysis.

**MSK_CALLBACK_IM_DUAL_SIMPLEX**

The call-back function is called at an intermediate point in the dual simplex optimizer.

**MSK_CALLBACK_IM_FULL_CONVEXITY_CHECK**

The call-back function is called at an intermediate stage of the full convexity check.

**MSK_CALLBACK_IM_INTPNT**

The call-back function is called at an intermediate stage within the interior-point optimizer where the information database has not been updated.

**MSK_CALLBACK_IM_LICENSE_WAIT**

MOSEK is waiting for a license.

**MSK_CALLBACK_IM_LU**

The call-back function is called from within the LU factorization procedure at an intermediate point.

**MSK_CALLBACK_IM_MIO**

    The call-back function is called at an intermediate point in the mixed-integer optimizer.

**MSK_CALLBACK_IM_MIO_DUAL_SIMPLEX**

    The call-back function is called at an intermediate point in the mixed-integer optimizer while running the dual simplex optimizer.

**MSK_CALLBACK_IM_MIO_INTPNT**

    The call-back function is called at an intermediate point in the mixed-integer optimizer while running the interior-point optimizer.

**MSK_CALLBACK_IM_MIO_PRESOLVE**

    The call-back function is called at an intermediate point in the mixed-integer optimizer while running the presolve.

**MSK_CALLBACK_IM_MIO_PRIMAL_SIMPLEX**

    The call-back function is called at an intermediate point in the mixed-integer optimizer while running the primal simplex optimizer.

**MSK_CALLBACK_IM_NETWORK_DUAL_SIMPLEX**

    The call-back function is called at an intermediate point in the dual network simplex optimizer.

**MSK_CALLBACK_IM_NETWORK_PRIMAL_SIMPLEX**

    The call-back function is called at an intermediate point in the primal network simplex optimizer.

**MSK_CALLBACK_IM_NONCONVEX**

    The call-back function is called at an intermediate stage within the nonconvex optimizer where the information database has not been updated.

**MSK_CALLBACK_IM_ORDER**

    The call-back function is called from within the matrix ordering procedure at an intermediate point.

**MSK_CALLBACK_IM_PRESOLVE**

    The call-back function is called from within the presolve procedure at an intermediate stage.

**MSK_CALLBACK_IM_PRIMAL_BI**

    The call-back function is called from within the basis identification procedure at an intermediate point in the primal phase.

**MSK_CALLBACK_IM_PRIMAL_DUAL_SIMPLEX**

    The call-back function is called at an intermediate point in the primal-dual simplex optimizer.

**MSK_CALLBACK_IM_PRIMAL_SENSIVITY**

    The call-back function is called at an intermediate stage of the primal sensitivity analysis.

MSK␣CALLBACK␣IM␣PRIMAL␣SIMPLEX

    The call-back function is called at an intermediate point in the primal simplex optimizer.

MSK␣CALLBACK␣IM␣QO␣REFORMULATE

    The call-back function is called at an intermediate stage of the conic quadratic reformulation.

MSK␣CALLBACK␣IM␣READ

    Intermediate stage in reading.

MSK␣CALLBACK␣IM␣SIMPLEX

    The call-back function is called from within the simplex optimizer at an intermediate point.

MSK␣CALLBACK␣IM␣SIMPLEX␣BI

    The call-back function is called from within the basis identification procedure at an intermediate point in the simplex clean-up phase. The frequency of the call-backs is controlled by the MSK␣IPAR␣LOG␣SIM␣FREQ parameter.

MSK␣CALLBACK␣INTPNT

    The call-back function is called from within the interior-point optimizer after the information database has been updated.

MSK␣CALLBACK␣NEW␣INT␣MIO

    The call-back function is called after a new integer solution has been located by the mixed-integer optimizer.

MSK␣CALLBACK␣NONCOVEX

    The call-back function is called from within the nonconvex optimizer after the information database has been updated.

MSK␣CALLBACK␣PRIMAL␣SIMPLEX

    The call-back function is called from within the primal simplex optimizer.

MSK␣CALLBACK␣READ␣OPF

    The call-back function is called from the OPF reader.

MSK␣CALLBACK␣READ␣OPF␣SECTION

    A chunk of $Q$ non-zeos has been read from a problem file.

MSK␣CALLBACK␣UPDATE␣DUAL␣BI

    The call-back function is called from within the basis identification procedure at an intermediate point in the dual phase.

MSK␣CALLBACK␣UPDATE␣DUAL␣SIMPLEX

    The call-back function is called in the dual simplex optimizer.

MSK_CALLBACK_UPDATE_DUAL_SIMPLEX_BI

>   The call-back function is called from within the basis identification procedure at an intermediate
>   point in the dual simplex clean-up phase. The frequency of the call-backs is controlled by the
>   MSK_IPAR_LOG_SIM_FREQ parameter.

MSK_CALLBACK_UPDATE_NETWORK_DUAL_SIMPLEX

>   The call-back function is called in the dual network simplex optimizer.

MSK_CALLBACK_UPDATE_NETWORK_PRIMAL_SIMPLEX

>   The call-back function is called in the primal network simplex optimizer.

MSK_CALLBACK_UPDATE_NONCONVEX

>   The call-back function is called at an intermediate stage within the nonconvex optimizer where
>   the information database has been updated.

MSK_CALLBACK_UPDATE_PRESOLVE

>   The call-back function is called from within the presolve procedure.

MSK_CALLBACK_UPDATE_PRIMAL_BI

>   The call-back function is called from within the basis identification procedure at an intermediate
>   point in the primal phase.

MSK_CALLBACK_UPDATE_PRIMAL_DUAL_SIMPLEX

>   The call-back function is called in the primal-dual simplex optimizer.

MSK_CALLBACK_UPDATE_PRIMAL_DUAL_SIMPLEX_BI

>   The call-back function is called from within the basis identification procedure at an intermediate
>   point in the primal-dual simplex clean-up phase. The frequency of the call-backs is controlled
>   by the MSK_IPAR_LOG_SIM_FREQ parameter.

MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX

>   The call-back function is called in the primal simplex optimizer.

MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX_BI

>   The call-back function is called from within the basis identification procedure at an intermediate
>   point in the primal simplex clean-up phase. The frequency of the call-backs is controlled by the
>   MSK_IPAR_LOG_SIM_FREQ parameter.

MSK_CALLBACK_WRITE_OPF

>   The call-back function is called from the OPF writer.

## 25.6   Types of convexity checks.

`MSK_CHECK_CONVEXITY_NONE`

> No convexity check.

`MSK_CHECK_CONVEXITY_SIMPLE`

> Perform simple and fast convexity check.

`MSK_CHECK_CONVEXITY_FULL`

> Perform a full convexity check.

## 25.7   Compression types

`MSK_COMPRESS_NONE`

> No compression is used.

`MSK_COMPRESS_FREE`

> The type of compression used is chosen automatically.

`MSK_COMPRESS_GZIP`

> The type of compression used is gzip compatible.

## 25.8   Cone types

`MSK_CT_QUAD`

> The cone is a quadratic cone.

`MSK_CT_RQUAD`

> The cone is a rotated quadratic cone.

## 25.9   Data format types

`MSK_DATA_FORMAT_EXTENSION`

> The file extension is used to determine the data file format.

`MSK_DATA_FORMAT_MPS`

> The data file is MPS formatted.

`MSK_DATA_FORMAT_LP`

> The data file is LP formatted.

**MSK_DATA_FORMAT_OP**

 The data file is an optimization problem formatted file.

**MSK_DATA_FORMAT_XML**

 The data file is an XML formatted file.

**MSK_DATA_FORMAT_FREE_MPS**

 The data data a free MPS formatted file.

**MSK_DATA_FORMAT_TASK**

 Generic task dump file.

**MSK_DATA_FORMAT_CB**

 Conic benchmark format.

## 25.10   Double information items

**MSK_DINF_BI_CLEAN_DUAL_TIME**

 Time spent within the dual clean-up optimizer of the basis identification procedure since its invocation.

**MSK_DINF_BI_CLEAN_PRIMAL_DUAL_TIME**

 Time spent within the primal-dual clean-up optimizer of the basis identification procedure since its invocation.

**MSK_DINF_BI_CLEAN_PRIMAL_TIME**

 Time spent within the primal clean-up optimizer of the basis identification procedure since its invocation.

**MSK_DINF_BI_CLEAN_TIME**

 Time spent within the clean-up phase of the basis identification procedure since its invocation.

**MSK_DINF_BI_DUAL_TIME**

 Time spent within the dual phase basis identification procedure since its invocation.

**MSK_DINF_BI_PRIMAL_TIME**

 Time spent within the primal phase of the basis identification procedure since its invocation.

**MSK_DINF_BI_TIME**

 Time spent within the basis identification procedure since its invocation.

**MSK_DINF_CONCURRENT_TIME**

 Time spent within the concurrent optimizer since its invocation.

MSK_DINF_INTPNT_DUAL_FEAS

Dual feasibility measure reported by the interior-point optimizer. (For the interior-point optimizer this measure does not directly related to the original problem because a homogeneous model is employed.)

MSK_DINF_INTPNT_DUAL_OBJ

Dual objective value reported by the interior-point optimizer.

MSK_DINF_INTPNT_FACTOR_NUM_FLOPS

An estimate of the number of flops used in the factorization.

MSK_DINF_INTPNT_OPT_STATUS

This measure should converge to +1 if the problem has a primal-dual optimal solution, and converge to -1 if problem is (strictly) primal or dual infeasible. Furthermore, if the measure converges to 0 the problem is usually ill-posed.

MSK_DINF_INTPNT_ORDER_TIME

Order time (in seconds).

MSK_DINF_INTPNT_PRIMAL_FEAS

Primal feasibility measure reported by the interior-point optimizers. (For the interior-point optimizer this measure does not directly related to the original problem because a homogeneous model is employed).

MSK_DINF_INTPNT_PRIMAL_OBJ

Primal objective value reported by the interior-point optimizer.

MSK_DINF_INTPNT_TIME

Time spent within the interior-point optimizer since its invocation.

MSK_DINF_MIO_CG_SEPERATION_TIME

Seperation time for CG cuts.

MSK_DINF_MIO_CMIR_SEPERATION_TIME

Seperation time for CMIR cuts.

MSK_DINF_MIO_CONSTRUCT_SOLUTION_OBJ

If MOSEK has successfully constructed an integer feasible solution, then this item contains the optimal objective value corresponding to the feasible solution.

MSK_DINF_MIO_DUAL_BOUND_AFTER_PRESOLVE

Value of the dual bound after presolve but before cut generation.

MSK_DINF_MIO_HEURISTIC_TIME

Time spent in the optimizer while solving the relaxtions.

MSK_DINF_MIO_OBJ_ABS_GAP

   Given the mixed-integer optimizer has computed a feasible solution and a bound on the optimal
   objective value, then this item contains the absolute gap defined by

$$|(\text{objective value of feasible solution}) - (\text{objective bound})|.$$

   Otherwise it has the value -1.0.

MSK_DINF_MIO_OBJ_BOUND

   The best known bound on the objective function. This value is undefined until at least one
   relaxation has been solved: To see if this is the case check that MSK_IINF_MIO_NUM_RELAX is
   stricly positive.

MSK_DINF_MIO_OBJ_INT

   The primal objective value corresponding to the best integer feasible solution. Please note that at
   least one integer feasible solution must have located i.e. check MSK_IINF_MIO_NUM_INT_SOLUTIONS.

MSK_DINF_MIO_OBJ_REL_GAP

   Given that the mixed-integer optimizer has computed a feasible solution and a bound on the
   optimal objective value, then this item contains the relative gap defined by

$$\frac{|(\text{objective value of feasible solution}) - (\text{objective bound})|}{\max(\delta, |(\text{objective value of feasible solution})|)}.$$

   where $\delta$ is given by the paramater MSK_DPAR_MIO_REL_GAP_CONST. Otherwise it has the value -1.0.

MSK_DINF_MIO_OPTIMIZER_TIME

   Time spent in the optimizer while solving the relaxtions.

MSK_DINF_MIO_PROBING_TIME

   Total time for probing.

MSK_DINF_MIO_ROOT_CUTGEN_TIME

   Total time for cut generation.

MSK_DINF_MIO_ROOT_OPTIMIZER_TIME

   Time spent in the optimizer while solving the root relaxation.

MSK_DINF_MIO_ROOT_PRESOLVE_TIME

   Time spent in while presolveing the root relaxation.

MSK_DINF_MIO_TIME

   Time spent in the mixed-integer optimizer.

MSK_DINF_MIO_USER_OBJ_CUT

   If the objective cut is used, then this information item has the value of the cut.

MSK_DINF_OPTIMIZER_TIME

Total time spent in the optimizer since it was invoked.

MSK_DINF_PRESOLVE_ELI_TIME

Total time spent in the eliminator since the presolve was invoked.

MSK_DINF_PRESOLVE_LINDEP_TIME

Total time spent in the linear dependency checker since the presolve was invoked.

MSK_DINF_PRESOLVE_TIME

Total time (in seconds) spent in the presolve since it was invoked.

MSK_DINF_PRIMAL_REPAIR_PENALTY_OBJ

The optimal objective value of the penalty function.

MSK_DINF_QCQO_REFORMULATE_TIME

Time spent with conic quadratic reformulation.

MSK_DINF_RD_TIME

Time spent reading the data file.

MSK_DINF_SIM_DUAL_TIME

Time spent in the dual simplex optimizer since invoking it.

MSK_DINF_SIM_FEAS

Feasibility measure reported by the simplex optimizer.

MSK_DINF_SIM_NETWORK_DUAL_TIME

Time spent in the dual network simplex optimizer since invoking it.

MSK_DINF_SIM_NETWORK_PRIMAL_TIME

Time spent in the primal network simplex optimizer since invoking it.

MSK_DINF_SIM_NETWORK_TIME

Time spent in the network simplex optimizer since invoking it.

MSK_DINF_SIM_OBJ

Objective value reported by the simplex optimizer.

MSK_DINF_SIM_PRIMAL_DUAL_TIME

Time spent in the primal-dual simplex optimizer optimizer since invoking it.

MSK_DINF_SIM_PRIMAL_TIME

Time spent in the primal simplex optimizer since invoking it.

MSK_DINF_SIM_TIME

Time spent in the simplex optimizer since invoking it.

MSK_DINF_SOL_BAS_DUAL_OBJ

> Dual objective value of the basic solution.

MSK_DINF_SOL_BAS_DVIOLCON

> Maximal dual bound violation for $x^c$ in the basic solution.

MSK_DINF_SOL_BAS_DVIOLVAR

> Maximal dual bound violation for $x^x$ in the basic solution.

MSK_DINF_SOL_BAS_PRIMAL_OBJ

> Primal objective value of the basic solution.

MSK_DINF_SOL_BAS_PVIOLCON

> Maximal primal bound violation for $x^c$ in the basic solution.

MSK_DINF_SOL_BAS_PVIOLVAR

> Maximal primal bound violation for $x^x$ in the basic solution.

MSK_DINF_SOL_ITG_PRIMAL_OBJ

> Primal objective value of the integer solution.

MSK_DINF_SOL_ITG_PVIOLBARVAR

> Maximal primal bound violation for $\bar{X}$ in the integer solution.

MSK_DINF_SOL_ITG_PVIOLCON

> Maximal primal bound violation for $x^c$ in the integer solution.

MSK_DINF_SOL_ITG_PVIOLCONES

> Maximal primal violation for primal conic constraints in the integer solution.

MSK_DINF_SOL_ITG_PVIOLITG

> Maximal violation for the integer constraints in the integer solution.

MSK_DINF_SOL_ITG_PVIOLVAR

> Maximal primal bound violation for $x^x$ in the integer solution.

MSK_DINF_SOL_ITR_DUAL_OBJ

> Dual objective value of the interior-point solution.

MSK_DINF_SOL_ITR_DVIOLBARVAR

> Maximal dual bound violation for $\bar{X}$ in the interior-point solution.

MSK_DINF_SOL_ITR_DVIOLCON

> Maximal dual bound violation for $x^c$ in the interior-point solution.

MSK_DINF_SOL_ITR_DVIOLCONES

> Maximal dual violation for dual conic constraints in the interior-point solution.

MSK_DINF_SOL_ITR_DVIOLVAR

    Maximal dual bound violation for $x^x$ in the interior-point solution.

MSK_DINF_SOL_ITR_PRIMAL_OBJ

    Primal objective value of the interior-point solution.

MSK_DINF_SOL_ITR_PVIOLBARVAR

    Maximal primal bound violation for $\bar{X}$ in the interior-point solution.

MSK_DINF_SOL_ITR_PVIOLCON

    Maximal primal bound violation for $x^c$ in the interior-point solution.

MSK_DINF_SOL_ITR_PVIOLCONES

    Maximal primal violation for primal conic constraints in the interior-point solution.

MSK_DINF_SOL_ITR_PVIOLVAR

    Maximal primal bound violation for $x^x$ in the interior-point solution.

## 25.11 Feasibility repair types

MSK_FEASREPAIR_OPTIMIZE_NONE

    Do not optimize the feasibility repair problem.

MSK_FEASREPAIR_OPTIMIZE_PENALTY

    Minimize weighted sum of violations.

MSK_FEASREPAIR_OPTIMIZE_COMBINED

    Minimize with original objective subject to minimal weighted violation of bounds.

## 25.12 License feature

MSK_FEATURE_PTS

    Base system.

MSK_FEATURE_PTON

    Nonlinear extension.

MSK_FEATURE_PTOM

    Mixed-integer extension.

MSK_FEATURE_PTOX

    Non-convex extension.

## 25.13    Integer information items.

MSK_IINF_ANA_PRO_NUM_CON

> Number of constraints in the problem.

MSK_IINF_ANA_PRO_NUM_CON_EQ

> Number of equality constraints.

MSK_IINF_ANA_PRO_NUM_CON_FR

> Number of unbounded constraints.

MSK_IINF_ANA_PRO_NUM_CON_LO

> Number of constraints with a lower bound and an infinite upper bound.

MSK_IINF_ANA_PRO_NUM_CON_RA

> Number of constraints with finite lower and upper bounds.

MSK_IINF_ANA_PRO_NUM_CON_UP

> Number of constraints with an upper bound and an infinite lower bound.

MSK_IINF_ANA_PRO_NUM_VAR

> Number of variables in the problem.

MSK_IINF_ANA_PRO_NUM_VAR_BIN

> Number of binary (0-1) variables.

MSK_IINF_ANA_PRO_NUM_VAR_CONT

> Number of continuous variables.

MSK_IINF_ANA_PRO_NUM_VAR_EQ

> Number of fixed variables.

MSK_IINF_ANA_PRO_NUM_VAR_FR

> Number of free variables.

MSK_IINF_ANA_PRO_NUM_VAR_INT

> Number of general integer variables.

MSK_IINF_ANA_PRO_NUM_VAR_LO

> Number of variables with a lower bound and an infinite upper bound.

MSK_IINF_ANA_PRO_NUM_VAR_RA

> Number of variables with finite lower and upper bounds.

MSK_IINF_ANA_PRO_NUM_VAR_UP

> Number of variables with an upper bound and an infinite lower bound. This value is set by

MSK_IINF_CONCURRENT_FASTEST_OPTIMIZER

> The type of the optimizer that finished first in a concurrent optimization.

MSK_IINF_INTPNT_FACTOR_DIM_DENSE

> Dimension of the dense sub system in factorization.

MSK_IINF_INTPNT_ITER

> Number of interior-point iterations since invoking the interior-point optimizer.

MSK_IINF_INTPNT_NUM_THREADS

> Number of threads that the interior-point optimizer is using.

MSK_IINF_INTPNT_SOLVE_DUAL

> Non-zero if the interior-point optimizer is solving the dual problem.

MSK_IINF_MIO_CONSTRUCT_NUM_ROUNDINGS

> Number of values in the integer solution that is rounded to an integer value.

MSK_IINF_MIO_CONSTRUCT_SOLUTION

> If this item has the value 0, then MOSEK did not try to construct an initial integer feasible solution. If the item has a positive value, then MOSEK successfully constructed an initial integer feasible solution.

MSK_IINF_MIO_INITIAL_SOLUTION

> Is non-zero if an initial integer solution is specified.

MSK_IINF_MIO_NUM_ACTIVE_NODES

> Number of active brabch bound nodes.

MSK_IINF_MIO_NUM_BASIS_CUTS

> Number of basis cuts.

MSK_IINF_MIO_NUM_BRANCH

> Number of branches performed during the optimization.

MSK_IINF_MIO_NUM_CARDGUB_CUTS

> Number of cardgub cuts.

MSK_IINF_MIO_NUM_CLIQUE_CUTS

> Number of clique cuts.

MSK_IINF_MIO_NUM_COEF_REDC_CUTS

> Number of coef. redc. cuts.

MSK_IINF_MIO_NUM_CONTRA_CUTS

> Number of contra cuts.

MSK_IINF_MIO_NUM_DISAGG_CUTS

    Number of diasagg cuts.

MSK_IINF_MIO_NUM_FLOW_COVER_CUTS

    Number of flow cover cuts.

MSK_IINF_MIO_NUM_GCD_CUTS

    Number of gcd cuts.

MSK_IINF_MIO_NUM_GOMORY_CUTS

    Number of Gomory cuts.

MSK_IINF_MIO_NUM_GUB_COVER_CUTS

    Number of GUB cover cuts.

MSK_IINF_MIO_NUM_INT_SOLUTIONS

    Number of integer feasible solutions that has been found.

MSK_IINF_MIO_NUM_KNAPSUR_COVER_CUTS

    Number of knapsack cover cuts.

MSK_IINF_MIO_NUM_LATTICE_CUTS

    Number of lattice cuts.

MSK_IINF_MIO_NUM_LIFT_CUTS

    Number of lift cuts.

MSK_IINF_MIO_NUM_OBJ_CUTS

    Number of obj cuts.

MSK_IINF_MIO_NUM_PLAN_LOC_CUTS

    Number of loc cuts.

MSK_IINF_MIO_NUM_RELAX

    Number of relaxations solved during the optimization.

MSK_IINF_MIO_NUMCON

    Number of constraints in the problem solved be the mixed-integer optimizer.

MSK_IINF_MIO_NUMINT

    Number of integer variables in the problem solved be the mixed-integer optimizer.

MSK_IINF_MIO_NUMVAR

    Number of variables in the problem solved be the mixed-integer optimizer.

MSK_IINF_MIO_OBJ_BOUND_DEFINED

    Non-zero if a valid objective bound has been found, otherwise zero.

MSK_IINF_MIO_TOTAL_NUM_CUTS

    Total number of cuts generated by the mixed-integer optimizer.

MSK_IINF_MIO_USER_OBJ_CUT

    If it is non-zero, then the objective cut is used.

MSK_IINF_OPT_NUMCON

    Number of constraints in the problem solved when the optimizer is called.

MSK_IINF_OPT_NUMVAR

    Number of variables in the problem solved when the optimizer is called

MSK_IINF_OPTIMIZE_RESPONSE

    The reponse code returned by optimize.

MSK_IINF_RD_NUMBARVAR

    Number of variables read.

MSK_IINF_RD_NUMCON

    Number of constraints read.

MSK_IINF_RD_NUMCONE

    Number of conic constraints read.

MSK_IINF_RD_NUMINTVAR

    Number of integer-constrained variables read.

MSK_IINF_RD_NUMQ

    Number of nonempty Q matrixes read.

MSK_IINF_RD_NUMVAR

    Number of variables read.

MSK_IINF_RD_PROTYPE

    Problem type.

MSK_IINF_SIM_DUAL_DEG_ITER

    The number of dual degenerate iterations.

MSK_IINF_SIM_DUAL_HOTSTART

    If 1 then the dual simplex algorithm is solving from an advanced basis.

MSK_IINF_SIM_DUAL_HOTSTART_LU

    If 1 then a valid basis factorization of full rank was located and used by the dual simplex algorithm.

MSK_IINF_SIM_DUAL_INF_ITER

>   The number of iterations taken with dual infeasibility.

MSK_IINF_SIM_DUAL_ITER

>   Number of dual simplex iterations during the last optimization.

MSK_IINF_SIM_NETWORK_DUAL_DEG_ITER

>   The number of dual network degenerate iterations.

MSK_IINF_SIM_NETWORK_DUAL_HOTSTART

>   If 1 then the dual network simplex algorithm is solving from an advanced basis.

MSK_IINF_SIM_NETWORK_DUAL_HOTSTART_LU

>   If 1 then a valid basis factorization of full rank was located and used by the dual network simplex
>   algorithm.

MSK_IINF_SIM_NETWORK_DUAL_INF_ITER

>   The number of iterations taken with dual infeasibility in the network optimizer.

MSK_IINF_SIM_NETWORK_DUAL_ITER

>   Number of dual network simplex iterations during the last optimization.

MSK_IINF_SIM_NETWORK_PRIMAL_DEG_ITER

>   The number of primal network degenerate iterations.

MSK_IINF_SIM_NETWORK_PRIMAL_HOTSTART

>   If 1 then the primal network simplex algorithm is solving from an advanced basis.

MSK_IINF_SIM_NETWORK_PRIMAL_HOTSTART_LU

>   If 1 then a valid basis factorization of full rank was located and used by the primal network
>   simplex algorithm.

MSK_IINF_SIM_NETWORK_PRIMAL_INF_ITER

>   The number of iterations taken with primal infeasibility in the network optimizer.

MSK_IINF_SIM_NETWORK_PRIMAL_ITER

>   Number of primal network simplex iterations during the last optimization.

MSK_IINF_SIM_NUMCON

>   Number of constraints in the problem solved by the simplex optimizer.

MSK_IINF_SIM_NUMVAR

>   Number of variables in the problem solved by the simplex optimizer.

MSK_IINF_SIM_PRIMAL_DEG_ITER

>   The number of primal degenerate iterations.

MSK_IINF_SIM_PRIMAL_DUAL_DEG_ITER

The number of degenerate major iterations taken by the primal dual simplex algorithm.

MSK_IINF_SIM_PRIMAL_DUAL_HOTSTART

If 1 then the primal dual simplex algorithm is solving from an advanced basis.

MSK_IINF_SIM_PRIMAL_DUAL_HOTSTART_LU

If 1 then a valid basis factorization of full rank was located and used by the primal dual simplex algorithm.

MSK_IINF_SIM_PRIMAL_DUAL_INF_ITER

The number of master iterations with dual infeasibility taken by the primal dual simplex algorithm.

MSK_IINF_SIM_PRIMAL_DUAL_ITER

Number of primal dual simplex iterations during the last optimization.

MSK_IINF_SIM_PRIMAL_HOTSTART

If 1 then the primal simplex algorithm is solving from an advanced basis.

MSK_IINF_SIM_PRIMAL_HOTSTART_LU

If 1 then a valid basis factorization of full rank was located and used by the primal simplex algorithm.

MSK_IINF_SIM_PRIMAL_INF_ITER

The number of iterations taken with primal infeasibility.

MSK_IINF_SIM_PRIMAL_ITER

Number of primal simplex iterations during the last optimization.

MSK_IINF_SIM_SOLVE_DUAL

Is non-zero if dual problem is solved.

MSK_IINF_SOL_BAS_PROSTA

Problem status of the basic solution. Updated after each optimization.

MSK_IINF_SOL_BAS_SOLSTA

Solution status of the basic solution. Updated after each optimization.

MSK_IINF_SOL_INT_PROSTA

Deprecated.

MSK_IINF_SOL_INT_SOLSTA

Degrecated.

MSK_IINF_SOL_ITG_PROSTA

Problem status of the integer solution. Updated after each optimization.

**MSK_IINF_SOL_ITG_SOLSTA**

> Solution status of the integer solution. Updated after each optimization.

**MSK_IINF_SOL_ITR_PROSTA**

> Problem status of the interior-point solution. Updated after each optimization.

**MSK_IINF_SOL_ITR_SOLSTA**

> Solution status of the interior-point solution. Updated after each optimization.

**MSK_IINF_STO_NUM_A_CACHE_FLUSHES**

> Number of times the cache of $A$ elements is flushed. A large number implies that `maxnumanz` is too small as well as an inefficient usage of MOSEK.

**MSK_IINF_STO_NUM_A_REALLOC**

> Number of times the storage for storing $A$ has been changed. A large value may indicates that memory fragmentation may occur.

**MSK_IINF_STO_NUM_A_TRANSPOSES**

> Number of times the $A$ matrix is transposed. A large number implies that `maxnumanz` is too small or an inefficient usage of MOSEK. This will occur in particular if the code alternate between accessing rows and columns of $A$.

## 25.14   Information item types

**MSK_INF_DOU_TYPE**

> Is a double information type.

**MSK_INF_INT_TYPE**

> Is an integer.

**MSK_INF_LINT_TYPE**

> Is a long integer.

## 25.15   Hot-start type employed by the interior-point optimizers.

**MSK_INTPNT_HOTSTART_NONE**

> The interior-point optimizer performs a coldstart.

**MSK_INTPNT_HOTSTART_PRIMAL**

> The interior-point optimizer exploits the primal solution only.

MSK␣INTPNT␣HOTSTART␣DUAL

> The interior-point optimizer exploits the dual solution only.

MSK␣INTPNT␣HOTSTART␣PRIMAL␣DUAL

> The interior-point optimizer exploits both the primal and dual solution.

## 25.16 Input/output modes

MSK␣IOMODE␣READ

> The file is read-only.

MSK␣IOMODE␣WRITE

> The file is write-only. If the file exists then it is truncated when it is opened. Otherwise it is created when it is opened.

MSK␣IOMODE␣READWRITE

> The file is to read and written.

## 25.17 Language selection constants

MSK␣LANG␣ENG

> English language selection

MSK␣LANG␣DAN

> Danish language selection

## 25.18 Long integer information items.

MSK␣LIINF␣BI␣CLEAN␣DUAL␣DEG␣ITER

> Number of dual degenerate clean iterations performed in the basis identification.

MSK␣LIINF␣BI␣CLEAN␣DUAL␣ITER

> Number of dual clean iterations performed in the basis identification.

MSK␣LIINF␣BI␣CLEAN␣PRIMAL␣DEG␣ITER

> Number of primal degenerate clean iterations performed in the basis identification.

MSK␣LIINF␣BI␣CLEAN␣PRIMAL␣DUAL␣DEG␣ITER

> Number of primal-dual degenerate clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_PRIMAL_DUAL_ITER

> Number of primal-dual clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_PRIMAL_DUAL_SUB_ITER

> Number of primal-dual subproblem clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_PRIMAL_ITER

> Number of primal clean iterations performed in the basis identification.

MSK_LIINF_BI_DUAL_ITER

> Number of dual pivots performed in the basis identification.

MSK_LIINF_BI_PRIMAL_ITER

> Number of primal pivots performed in the basis identification.

MSK_LIINF_INTPNT_FACTOR_NUM_NZ

> Number of non-zeros in factorization.

MSK_LIINF_MIO_INTPNT_ITER

> Number of interior-point iterations performed by the mixed-integer optimizer.

MSK_LIINF_MIO_SIMPLEX_ITER

> Number of simplex iterations performed by the mixed-integer optimizer.

MSK_LIINF_RD_NUMANZ

> Number of non-zeros in A that is read.

MSK_LIINF_RD_NUMQNZ

> Number of Q non-zeros.

## 25.19   Mark

MSK_MARK_LO

> The lower bound is selected for sensitivity analysis.

MSK_MARK_UP

> The upper bound is selected for sensitivity analysis.

## 25.20   Continuous mixed-integer solution type

MSK_MIO_CONT_SOL_NONE

    No interior-point or basic solution are reported when the mixed-integer optimizer is used.

MSK_MIO_CONT_SOL_ROOT

    The reported interior-point and basic solutions are a solution to the root node problem when mixed-integer optimizer is used.

MSK_MIO_CONT_SOL_ITG

    The reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. A solution is only reported in case the problem has a primal feasible solution.

MSK_MIO_CONT_SOL_ITG_REL

    In case the problem is primal feasible then the reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. If the problem is primal infeasible, then the solution to the root node problem is reported.

## 25.21   Integer restrictions

MSK_MIO_MODE_IGNORED

    The integer constraints are ignored and the problem is solved as a continuous problem.

MSK_MIO_MODE_SATISFIED

    Integer restrictions should be satisfied.

MSK_MIO_MODE_LAZY

    Integer restrictions should be satisfied if an optimizer is available for the problem.

## 25.22   Mixed-integer node selection types

MSK_MIO_NODE_SELECTION_FREE

    The optimizer decides the node selection strategy.

MSK_MIO_NODE_SELECTION_FIRST

    The optimizer employs a depth first node selection strategy.

MSK_MIO_NODE_SELECTION_BEST

    The optimizer employs a best bound node selection strategy.

**MSK_MIO_NODE_SELECTION_WORST**

   The optimizer employs a worst bound node selection strategy.

**MSK_MIO_NODE_SELECTION_HYBRID**

   The optimizer employs a hybrid strategy.

**MSK_MIO_NODE_SELECTION_PSEUDO**

   The optimizer employs selects the node based on a pseudo cost estimate.

## 25.23   MPS file format type

**MSK_MPS_FORMAT_STRICT**

   It is assumed that the input file satisfies the MPS format strictly.

**MSK_MPS_FORMAT_RELAXED**

   It is assumed that the input file satisfies a slightly relaxed version of the MPS format.

**MSK_MPS_FORMAT_FREE**

   It is assumed that the input file satisfies the free MPS format. This implies that spaces are not allowed in names. Otherwise the format is free.

## 25.24   Message keys

**MSK_MSG_READING_FILE**

**MSK_MSG_WRITING_FILE**

**MSK_MSG_MPS_SELECTED**

## 25.25   Name types

**MSK_NAME_TYPE_GEN**

   General names. However, no duplicate and blank names are allowed.

**MSK_NAME_TYPE_MPS**

   MPS type names.

**MSK_NAME_TYPE_LP**

   LP type names.

## 25.26   Objective sense types

**MSK_OBJECTIVE_SENSE_MINIMIZE**

> The problem should be minimized.

**MSK_OBJECTIVE_SENSE_MAXIMIZE**

> The problem should be maximized.

## 25.27   On/off

**MSK_OFF**

> Switch the option off.

**MSK_ON**

> Switch the option on.

## 25.28   Optimizer types

**MSK_OPTIMIZER_FREE**

> The optimizer is chosen automatically.

**MSK_OPTIMIZER_INTPNT**

> The interior-point optimizer is used.

**MSK_OPTIMIZER_CONIC**

> The optimizer for problems having conic constraints.

**MSK_OPTIMIZER_PRIMAL_SIMPLEX**

> The primal simplex optimizer is used.

**MSK_OPTIMIZER_DUAL_SIMPLEX**

> The dual simplex optimizer is used.

**MSK_OPTIMIZER_PRIMAL_DUAL_SIMPLEX**

> The primal dual simplex optimizer is used.

**MSK_OPTIMIZER_FREE_SIMPLEX**

> One of the simplex optimizers is used.

**MSK_OPTIMIZER_NETWORK_PRIMAL_SIMPLEX**

The network primal simplex optimizer is used. It is only applicable to pure network problems.

`MSK_OPTIMIZER_MIXED_INT_CONIC`

> The mixed-integer optimizer for conic and linear problems.

`MSK_OPTIMIZER_MIXED_INT`

> The mixed-integer optimizer.

`MSK_OPTIMIZER_CONCURRENT`

> The optimizer for nonconvex nonlinear problems.

`MSK_OPTIMIZER_NONCONVEX`

> The optimizer for nonconvex nonlinear problems.

## 25.29   Ordering strategies

`MSK_ORDER_METHOD_FREE`

> The ordering method is chosen automatically.

`MSK_ORDER_METHOD_APPMINLOC`

> Approximate minimum local fill-in ordering is employed.

`MSK_ORDER_METHOD_EXPERIMENTAL`

> This option should not be used.

`MSK_ORDER_METHOD_TRY_GRAPHPAR`

> Always try the the graph partitioning based ordering.

`MSK_ORDER_METHOD_FORCE_GRAPHPAR`

> Always use the graph partitioning based ordering even if it is worse that the approximate minimum local fill ordering.

`MSK_ORDER_METHOD_NONE`

> No ordering is used.

## 25.30   Parameter type

`MSK_PAR_INVALID_TYPE`

> Not a valid parameter.

`MSK_PAR_DOU_TYPE`

> Is a double parameter.

`MSK_PAR_INT_TYPE`

> Is an integer parameter.

MSK_PAR_STR_TYPE
> Is a string parameter.

## 25.31 Presolve method.

MSK_PRESOLVE_MODE_OFF
> The problem is not presolved before it is optimized.

MSK_PRESOLVE_MODE_ON
> The problem is presolved before it is optimized.

MSK_PRESOLVE_MODE_FREE
> It is decided automatically whether to presolve before the problem is optimized.

## 25.32 Problem data items

MSK_PI_VAR
> Item is a variable.

MSK_PI_CON
> Item is a constraint.

MSK_PI_CONE
> Item is a cone.

## 25.33 Problem types

MSK_PROBTYPE_LO
> The problem is a linear optimization problem.

MSK_PROBTYPE_QO
> The problem is a quadratic optimization problem.

MSK_PROBTYPE_QCQO
> The problem is a quadratically constrained optimization problem.

MSK_PROBTYPE_GECO
> General convex optimization.

MSK_PROBTYPE_CONIC
> A conic optimization.

MSK‗PROBTYPE‗MIXED

> General nonlinear constraints and conic constraints. This combination can not be solved by MOSEK.

## 25.34   Problem status keys

MSK‗PRO‗STA‗UNKNOWN

> Unknown problem status.

MSK‗PRO‗STA‗PRIM‗AND‗DUAL‗FEAS

> The problem is primal and dual feasible.

MSK‗PRO‗STA‗PRIM‗FEAS

> The problem is primal feasible.

MSK‗PRO‗STA‗DUAL‗FEAS

> The problem is dual feasible.

MSK‗PRO‗STA‗PRIM‗INFEAS

> The problem is primal infeasible.

MSK‗PRO‗STA‗DUAL‗INFEAS

> The problem is dual infeasible.

MSK‗PRO‗STA‗PRIM‗AND‗DUAL‗INFEAS

> The problem is primal and dual infeasible.

MSK‗PRO‗STA‗ILL‗POSED

> The problem is ill-posed. For example, it may be primal and dual feasible but have a positive duality gap.

MSK‗PRO‗STA‗NEAR‗PRIM‗AND‗DUAL‗FEAS

> The problem is at least nearly primal and dual feasible.

MSK‗PRO‗STA‗NEAR‗PRIM‗FEAS

> The problem is at least nearly primal feasible.

MSK‗PRO‗STA‗NEAR‗DUAL‗FEAS

> The problem is at least nearly dual feasible.

MSK‗PRO‗STA‗PRIM‗INFEAS‗OR‗UNBOUNDED

> The problem is either primal infeasible or unbounded. This may occur for mixed-integer problems.

## 25.35   Response code type

**MSK_RESPONSE_OK**

>   The response code is OK.

**MSK_RESPONSE_WRN**

>   The response code is a warning.

**MSK_RESPONSE_TRM**

>   The response code is an optimizer termination status.

**MSK_RESPONSE_ERR**

>   The response code is an error.

**MSK_RESPONSE_UNK**

>   The response code does not belong to any class.

## 25.36   Scaling type

**MSK_SCALING_METHOD_POW2**

>   Scales only with power of 2 leaving the mantissa untouched.

**MSK_SCALING_METHOD_FREE**

>   The optimizer chooses the scaling heuristic.

## 25.37   Scaling type

**MSK_SCALING_FREE**

>   The optimizer chooses the scaling heuristic.

**MSK_SCALING_NONE**

>   No scaling is performed.

**MSK_SCALING_MODERATE**

>   A conservative scaling is performed.

**MSK_SCALING_AGGRESSIVE**

>   A very aggressive scaling is performed.

## 25.38   Sensitivity types

`MSK_SENSITIVITY_TYPE_BASIS`

>   Basis sensitivity analysis is performed.

`MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION`

>   Optimal partition sensitivity analysis is performed.

## 25.39   Degeneracy strategies

`MSK_SIM_DEGEN_NONE`

>   The simplex optimizer should use no degeneration strategy.

`MSK_SIM_DEGEN_FREE`

>   The simplex optimizer chooses the degeneration strategy.

`MSK_SIM_DEGEN_AGGRESSIVE`

>   The simplex optimizer should use an aggressive degeneration strategy.

`MSK_SIM_DEGEN_MODERATE`

>   The simplex optimizer should use a moderate degeneration strategy.

`MSK_SIM_DEGEN_MINIMUM`

>   The simplex optimizer should use a minimum degeneration strategy.

## 25.40   Exploit duplicate columns.

`MSK_SIM_EXPLOIT_DUPVEC_OFF`

>   Disallow the simplex optimizer to exploit duplicated columns.

`MSK_SIM_EXPLOIT_DUPVEC_ON`

>   Allow the simplex optimizer to exploit duplicated columns.

`MSK_SIM_EXPLOIT_DUPVEC_FREE`

>   The simplex optimizer can choose freely.

## 25.41   Hot-start type employed by the simplex optimizer

`MSK_SIM_HOTSTART_NONE`

>   The simplex optimizer performs a coldstart.

MSK_SIM_HOTSTART_FREE

> The simplex optimize chooses the hot-start type.

MSK_SIM_HOTSTART_STATUS_KEYS

> Only the status keys of the constraints and variables are used to choose the type of hot-start.

## 25.42 Problem reformulation.

MSK_SIM_REFORMULATION_OFF

> Disallow the simplex optimizer to reformulate the problem.

MSK_SIM_REFORMULATION_ON

> Allow the simplex optimizer to reformulate the problem.

MSK_SIM_REFORMULATION_FREE

> The simplex optimizer can choose freely.

MSK_SIM_REFORMULATION_AGGRESSIVE

> The simplex optimizer should use an aggressive reformulation strategy.

## 25.43 Simplex selection strategy

MSK_SIM_SELECTION_FREE

> The optimizer chooses the pricing strategy.

MSK_SIM_SELECTION_FULL

> The optimizer uses full pricing.

MSK_SIM_SELECTION_ASE

> The optimizer uses approximate steepest-edge pricing.

MSK_SIM_SELECTION_DEVEX

> The optimizer uses devex steepest-edge pricing (or if it is not available an approximate steep-edge selection).

MSK_SIM_SELECTION_SE

> The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

MSK_SIM_SELECTION_PARTIAL

> The optimizer uses a partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.

## 25.44    Solution items

**MSK_SOL_ITEM_XC**

>    Solution for the constraints.

**MSK_SOL_ITEM_XX**

>    Variable solution.

**MSK_SOL_ITEM_Y**

>    Lagrange multipliers for equations.

**MSK_SOL_ITEM_SLC**

>    Lagrange multipliers for lower bounds on the constraints.

**MSK_SOL_ITEM_SUC**

>    Lagrange multipliers for upper bounds on the constraints.

**MSK_SOL_ITEM_SLX**

>    Lagrange multipliers for lower bounds on the variables.

**MSK_SOL_ITEM_SUX**

>    Lagrange multipliers for upper bounds on the variables.

**MSK_SOL_ITEM_SNX**

>    Lagrange multipliers corresponding to the conic constraints on the variables.


## 25.45    Solution status keys

**MSK_SOL_STA_UNKNOWN**

>    Status of the solution is unknown.

**MSK_SOL_STA_OPTIMAL**

>    The solution is optimal.

**MSK_SOL_STA_PRIM_FEAS**

>    The solution is primal feasible.

**MSK_SOL_STA_DUAL_FEAS**

>    The solution is dual feasible.

**MSK_SOL_STA_PRIM_AND_DUAL_FEAS**

>    The solution is both primal and dual feasible.

**MSK_SOL_STA_PRIM_INFEAS_CER**

>    The solution is a certificate of primal infeasibility.

MSK_SOL_STA_DUAL_INFEAS_CER

> The solution is a certificate of dual infeasibility.

MSK_SOL_STA_NEAR_OPTIMAL

> The solution is nearly optimal.

MSK_SOL_STA_NEAR_PRIM_FEAS

> The solution is nearly primal feasible.

MSK_SOL_STA_NEAR_DUAL_FEAS

> The solution is nearly dual feasible.

MSK_SOL_STA_NEAR_PRIM_AND_DUAL_FEAS

> The solution is nearly both primal and dual feasible.

MSK_SOL_STA_NEAR_PRIM_INFEAS_CER

> The solution is almost a certificate of primal infeasibility.

MSK_SOL_STA_NEAR_DUAL_INFEAS_CER

> The solution is almost a certificate of dual infeasibility.

MSK_SOL_STA_INTEGER_OPTIMAL

> The primal solution is integer optimal.

MSK_SOL_STA_NEAR_INTEGER_OPTIMAL

> The primal solution is near integer optimal.

## 25.46 Solution types

MSK_SOL_ITR

> The interior solution.

MSK_SOL_BAS

> The basic solution.

MSK_SOL_ITG

> The integer solution.

## 25.47    Solve primal or dual form

`MSK_SOLVE_FREE`

    The optimizer is free to solve either the primal or the dual problem.

`MSK_SOLVE_PRIMAL`

    The optimizer should solve the primal problem.

`MSK_SOLVE_DUAL`

    The optimizer should solve the dual problem.

## 25.48    Status keys

`MSK_SK_UNK`

    The status for the constraint or variable is unknown.

`MSK_SK_BAS`

    The constraint or variable is in the basis.

`MSK_SK_SUPBAS`

    The constraint or variable is super basic.

`MSK_SK_LOW`

    The constraint or variable is at its lower bound.

`MSK_SK_UPR`

    The constraint or variable is at its upper bound.

`MSK_SK_FIX`

    The constraint or variable is fixed.

`MSK_SK_INF`

    The constraint or variable is infeasible in the bounds.

## 25.49    Starting point types

`MSK_STARTING_POINT_FREE`

    The starting point is chosen automatically.

`MSK_STARTING_POINT_GUESS`

    The optimizer guesses a starting point.

MSK_STARTING_POINT_CONSTANT

> The optimizer constructs a starting point by assigning a constant value to all primal and dual variables. This starting point is normally robust.

MSK_STARTING_POINT_SATISFY_BOUNDS

> The starting point is choosen to satisfy all the simple bounds on nonlinear variables. If this starting point is employed, then more care than usual should employed when choosing the bounds on the nonlinear variables. In particular very tight bounds should be avoided.

## 25.50   Stream types

MSK_STREAM_LOG

> Log stream. Contains the aggregated contents of all other streams. This means that a message written to any other stream will also be written to this stream.

MSK_STREAM_MSG

> Message stream. Log information relating to performance and progress of the optimization is written to this stream.

MSK_STREAM_ERR

> Error stream. Error messages are written to this stream.

MSK_STREAM_WRN

> Warning stream. Warning messages are written to this stream.

## 25.51   Symmetric matrix types

MSK_SYMMAT_TYPE_SPARSE

> Sparse symmetric matrix.

## 25.52   Transposed matrix.

MSK_TRANSPOSE_NO

> No transpose is applied.

MSK_TRANSPOSE_YES

> A transpose is applied.

## 25.53   Triangular part of a symmetric matrix.

MSK_UPLO_LO

> Lower part.

MSK_UPLO_UP

> Upper part

## 25.54   Integer values

MSK_LICENSE_BUFFER_LENGTH

> The length of a license key buffer.

MSK_MAX_STR_LEN

> Maximum string length allowed in MOSEK.

## 25.55   Variable types

MSK_VAR_TYPE_CONT

> Is a continuous variable.

MSK_VAR_TYPE_INT

> Is an integer variable.

## 25.56   XML writer output mode

MSK_WRITE_XML_MODE_ROW

> Write in row order.

MSK_WRITE_XML_MODE_COL

> Write in column order.

# Chapter 26

# Problem analyzer examples

This appendix presents a few examples of the output produced by the problem analyzer described in Section 14.1. The first two problems are taken from the MIPLIB 2003 collection, http://miplib.zib.de/.

## 26.1    air04

```
Analyzing the problem

Constraints                 Bounds                      Variables
 fixed   : all               ranged  : all               bin : all


-------------------------------------------------------------------------------


Objective, min cx
   range: min |c|: 31.0000      max |c|: 2258.00
 distrib:        |c|          vars
          [31, 100)          176
       [100, 1e+03)         8084
   [1e+03, 2.26e+03]         644


-------------------------------------------------------------------------------


Constraint matrix A has
       823 rows (constraints)
      8904 columns (variables)
     72965 (0.995703%) nonzero entries (coefficients)

Row nonzeros, A_i
   range: min A_i: 2 (0.0224618%)     max A_i: 368 (4.13297%)
 distrib:         A_i         rows      rows%         acc%
                   2            2        0.24         0.24
               [3, 7]          4        0.49         0.73
              [8, 15]         19        2.31         3.04
             [16, 31]         80        9.72        12.76
             [32, 63]        236       28.68        41.43
            [64, 127]        289       35.12        76.55
```

```
              [128, 255]         186         22.60        99.15
              [256, 368]           7          0.85       100.00


Column nonzeros, A|j
   range: min A|j: 2 (0.243013%)    max A|j: 15 (1.8226%)
   distrib:         A|j          cols         cols%         acc%
                      2           118          1.33          1.33
                 [3, 7]          2853         32.04         33.37
                [8, 15]          5933         66.63        100.00


A nonzeros, A(ij)
   range: all |A(ij)| = 1.00000


--------------------------------------------------------------------------------


Constraint bounds, lb <= Ax <= ub
   distrib:         |b|                 lbs              ubs
                 [1, 10]                823              823

Variable bounds, lb <= x <= ub
   distrib:         |b|                 lbs              ubs
                      0                8904
                 [1, 10]                                 8904


--------------------------------------------------------------------------------
```

# 26.2   arki001

```
Analyzing the problem

Constraints                Bounds                     Variables
 lower bd:         82       lower bd:         38        cont:         850
 upper bd:        946       fixed    :        353        bin :         415
 fixed    :        20       free     :          1        int :         123
                            ranged   :        996


--------------------------------------------------------------------------------


Objective, min cx
   range: all |c| in {0.00000, 1.00000}
   distrib:         |c|          vars
                      0          1387
                      1             1


--------------------------------------------------------------------------------


Constraint matrix A has
      1048 rows (constraints)
      1388 columns (variables)
     20439 (1.40511%) nonzero entries (coefficients)

Row nonzeros, A_i
   range: min A_i: 1 (0.0720461%)    max A_i: 1046 (75.3602%)
   distrib:         A_i          rows         rows%         acc%
                      1            29          2.77          2.77
```

```
                       2        476       45.42       48.19
                  [3, 7]         49        4.68       52.86
                 [8, 15]         56        5.34       58.21
                [16, 31]         64        6.11       64.31
                [32, 63]        373       35.59       99.90
            [1024, 1046]          1        0.10      100.00

Column nonzeros, A|j
   range: min A|j: 1 (0.0954198%)    max A|j: 29 (2.76718%)
   distrib:        A|j       cols       cols%        acc%
                     1        381       27.45       27.45
                     2         19        1.37       28.82
                  [3, 7]        38        2.74       31.56
                 [8, 15]       233       16.79       48.34
                [16, 29]       717       51.66      100.00

A nonzeros, A(ij)
   range: min |A(ij)|: 0.000200000     max |A(ij)|: 2.33067e+07
   distrib:      A(ij)     coeffs
      [0.0002, 0.001)        167
       [0.001, 0.01)       1049
        [0.01, 0.1)       4553
          [0.1, 1)        8840
           [1, 10)        3822
          [10, 100)        630
        [100, 1e+03)       267
       [1e+03, 1e+04)      699
       [1e+04, 1e+05)      291
       [1e+05, 1e+06)       83
       [1e+06, 1e+07)       19
     [1e+07, 2.33e+07]      19


--------------------------------------------------------------------------------


Constraint bounds, lb <= Ax <= ub
   distrib:        |b|            lbs          ubs
          [0.1, 1)                            386
           [1, 10)                             74
         [10, 100)              101            456
        [100, 1000)                             34
       [1000, 10000)                            15
      [100000, 1e+06]            1              1

Variable bounds, lb <= x <= ub
   distrib:        |b|            lbs          ubs
                0                974            323
       [0.001, 0.01)                            19
          [0.1, 1)                370            57
           [1, 10)                41            704
         [10, 100]                 2            246


--------------------------------------------------------------------------------
```

## 26.3    Problem with both linear and quadratic constraints

```
Analyzing the problem

Constraints                 Bounds                      Variables
 lower bd:        40         upper bd:         1          cont: all
 upper bd:       121         fixed   :       204
 fixed   :      5480         free    :      5600
 ranged  :       161         ranged  :        40


--------------------------------------------------------------------------------


Objective, maximize cx
   range: all |c| in {0.00000, 15.4737}
 distrib:         |c|          vars
                    0          5844
              15.4737             1


--------------------------------------------------------------------------------


Constraint matrix A has
      5802 rows (constraints)
      5845 columns (variables)
      6480 (0.0191079%) nonzero entries (coefficients)

Row nonzeros, A_i
   range: min A_i: 0 (0%)      max A_i: 3 (0.0513259%)
 distrib:         A_i          rows        rows%         acc%
                    0            80         1.38         1.38
                    1          5003        86.23        87.61
                    2           680        11.72        99.33
                    3            39         0.67       100.00
0/80 empty rows have quadratic terms

Column nonzeros, A|j
   range: min A|j: 0 (0%)      max A|j: 15 (0.258532%)
 distrib:         A|j          cols        cols%         acc%
                    0           204         3.49         3.49
                    1          5521        94.46        97.95
                    2            40         0.68        98.63
               [3, 7]           40         0.68        99.32
              [8, 15]           40         0.68       100.00
0/204 empty columns correspond to variables used in conic
 and/or quadratic expressions only

A nonzeros, A(ij)
   range: min |A(ij)|: 2.02410e-05      max |A(ij)|: 35.8400
 distrib:       A(ij)        coeffs
  [2.02e-05, 0.0001)           40
     [0.0001, 0.001)          118
      [0.001, 0.01)          305
       [0.01, 0.1)           176
          [0.1, 1)            40
           [1, 10)          5721
         [10, 35.8]           80


--------------------------------------------------------------------------------
```

```
Constraint bounds, lb <= Ax <= ub
 distrib:          |b|              lbs              ubs
                     0             5481             5600
        [1000, 10000)                                 1
       [10000, 100000)                2                1
       [1e+06, 1e+07)               78               40
       [1e+08, 1e+09]              120              120

Variable bounds, lb <= x <= ub
 distrib:          |b|              lbs              ubs
                     0              243              203
           [0.1, 1)                 1                1
       [1e+06, 1e+07)                                40
       [1e+11, 1e+12]                                 1


--------------------------------------------------------------------------------


Quadratic constraints: 121

Gradient nonzeros, Qx
   range: min Qx: 1 (0.0171086%)     max Qx: 2720 (46.5355%)
 distrib:          Qx          cons        cons%         acc%
                    1          40         33.06        33.06
          [64, 127]          80         66.12        99.17
        [2048, 2720]          1          0.83       100.00


--------------------------------------------------------------------------------
```

## 26.4   Problem with both linear and conic constraints

```
Analyzing the problem

Constraints               Bounds                    Variables
 upper bd:      3600       fixed   :     3601         cont: all
 fixed   :     21760       free    :    28802


--------------------------------------------------------------------------------


Objective, minimize cx
   range: all |c| in {0.00000, 1.00000}
 distrib:          |c|          vars
                    0         32402
                    1             1


--------------------------------------------------------------------------------


Constraint matrix A has
    25360 rows (constraints)
    32403 columns (variables)
    93339 (0.0113587%) nonzero entries (coefficients)

Row nonzeros, A_i
   range: min A_i: 1 (0.00308613%)     max A_i: 8 (0.0246891%)
```

```
distrib:        A_i        rows        rows%         acc%
                  1        3600        14.20        14.20
                  2       10803        42.60        56.79
             [3, 7]        3995        15.75        72.55
                  8        6962        27.45       100.00


Column nonzeros, A|j
  range: min A|j: 0 (0%)     max A|j: 61 (0.240536%)
 distrib:        A|j        cols        cols%         acc%
                  0        3602        11.12        11.12
                  1       10800        33.33        44.45
                  2        7200        22.22        66.67
             [3, 7]        7279        22.46        89.13
            [8, 15]        3521        10.87       100.00
           [32, 61]           1         0.00       100.00
3600/3602 empty columns correspond to variables used in conic
 and/or quadratic constraints only


A nonzeros, A(ij)
  range: min |A(ij)|: 0.00833333     max |A(ij)|: 1.00000
 distrib:      A(ij)      coeffs
    [0.00833, 0.01)       57280
       [0.01, 0.1)          59
          [0.1, 1]       36000


--------------------------------------------------------------------------------


Constraint bounds, lb <= Ax <= ub
 distrib:        |b|           lbs           ubs
                  0         21760         21760
            [0.1, 1]                       3600

Variable bounds, lb <= x <= ub
 distrib:        |b|           lbs           ubs
            [1, 10]         3601         3601


--------------------------------------------------------------------------------


--------------------------------------------------------------------------------


Rotated quadratic cones: 3600
                dim         RQCs
                  4        3600
```

# Bibliography

[1] Chvátal, V.. Linear programming, 1983. W.H. Freeman and Company

[2] Nazareth, J. L.. Computer Solution of Linear Programs, 1987. Oxford University Press, New York

[3] Bazaraa, M. S., Sherali, H. D. and Shetty, C. M.. Nonlinear programming: Theory and algorithms, 2 edition, 1993. John Wiley and Sons, New York

[4] Williams, H. P.. Model building in mathematical programming, 3 edition, 1993. John Wiley and Sons

[5] Wolsey, L. A.. Integer programming, 1998. John Wiley and Sons

[6] Ben-Tal, A. and Nemirovski, A.. Robust solutions of Linear Programming problems contaminated with uncertain data. Math. Programming 3:411-424

[7] Ben-Tal, A. and Nemirovski, A. Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications, 2001. SIAM

[8] Boyd, S.P., Kim, S.J., Vandenberghe, L. and Hassibi, A.. A Tutorial on Geometric Programming., 2004. ISL, Electrical Engineering Department, Stanford University, Stanford, CA. Available at http://www.stanford.edu/ boyd/gp_tutorial.html

[9] Beightler, C. and Phillips, D. T.. Applied geometric programming, 1976. John Wiley and Sons, New York

[10] Andersen, E. D. and Andersen, K. D.. Presolving in linear programming. Math. Programming 2:221-245

[11] Andersen, E. D., Gondzio, J., Mészáros, Cs. and Xu, X.. Implementation of interior point methods for large scale linear programming, Interior-point methods of mathematical programming p. 189-252, 1996. Kluwer Academic Publishers

[12] Erling D. Andersen. The homogeneous and self-dual model and algorithm for linear optimization. Technical report TR-1-2009, 2009. MOSEK ApS. http://www.mosek.com/fileadmin/reports/tech/homolo.pdf

[13] Andersen, E. D. and Ye, Y.. Combining interior-point and pivoting algorithms. Management Sci. December 12:1719-1731

[14] Ahuja, R. K., Magnanti, T. L. and Orlin, J. B.. Network flows, Optimization, vol. 1 p. 211-369, 1989. North Holland, Amsterdam

[15] Andersen, E. D., Roos, C. and Terlaky, T.. On implementing a primal-dual interior-point method for conic quadratic optimization. Math. Programming February 2

[16] Andersen, E. D. and Ye, Y.. A computational study of the homogeneous algorithm for large-scale convex optimization. Computational Optimization and Applications 10:243-269

[17] Andersen, E. D. and Ye, Y.. On a homogeneous algorithm for the monotone complementarity problem. Math. Programming February 2:375-399

[18] Roos, C., Terlaky, T. and Vial, J. -Ph.. Theory and algorithms for linear optimization: an interior point approach, 1997. John Wiley and Sons, New York

[19] Wallace, S. W.. Decision making under uncertainty: Is sensitivity of any use. Oper. Res. January 1:20-25

[20] MOSEK ApS. MOSEK Modeling manual, 2012. Last revised January 31 2013. http://docs.mosek.com/generic/modeling-a4.pdf

# Index