The MOSEK optimization toolbox for MATLAB manual.

Version 5.0 (Revision 138).

www.mosek.com

Published by MOSEK ApS, Denmark.

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 Ø
Denmark

Contents

License agreement

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

`mosek\5\license\index.html`

Chapter 1

Changes and new features in MOSEK

The section presents improvements and new features added to MOSEK in version 5.0.

1.1 File formats

- The `OSiL` XML format for linear problems is now supported as output-only format.
- The new Optimization Problem file Format (OPF) is now available. It incorporates linear, quadratic, and conic problems in a single format, as well as parameter settings and solutions.
- The `OBJNAME` section is now supported in the MPS format.

1.2 Optimizers

- The interior-point solver is about 20% faster on average for large linear problems, compared to MOSEK 4.0.
- The dual simplex solver is about 40% faster on average compared to MOSEK 4.0.
- For the primal simplex solver, handling of problems with long slim structure has been improved.
- For both simplex optimizers numerical stability, hot-start efficiency and degeneracy handling has been improved substantially.
- A simplex network flow optimizer is now available. In many cases the specialized simplex optimizer can solve a pure network flow optimization problem up to 10 times faster than the standard simplex optimizer.

- Presolve is now by default turned on for hot-start with the simplex optimizers.
- The mixed integer optimizer now includes the feasibility pump heuristic to find a good initial feasible solution.
- Full support for setting branching priorities on integer constrained variables.

1.3 API changes

- The function `MSK_putobjsense` has been introduced. This should be used to define objective sense instead of the parameter `MSK_IPAR_OBJECTIVE_SENSE`.

1.4 License system

- The Flexlm license software has been upgraded to version 11.4.
- Dongles are supported in 64 bit Windows.

1.5 Other changes

- The documentation has been improved. Each interface now have a complete dedicated manual, and many code examples have been added. The HTML version has been subject to heavy cosmetical changes.

1.6 Interfaces

- A complete Python interface is now available.
- The MATLAB interface supports the MATLAB versions R2006a, R2006b, and R2007a.
- The general convex interface has been disabled in the Java and .NET interfaces.
- The Java API provides an interface to the native `scopt` functionality.

1.7 Supported platforms

- Mac OSX 32 bit for x86 version has been added.
- Solaris 32 bit for x86 version has been added
- Solaris 64 bit for x86 version has been added.

Chapter 2

Introduction

This manual describe 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 communicated to the optimization software. Finally, when the results has been obtained, results are analyzed and interpreted. A popular software tool for these tasks is MATLAB[1]. The MOSEK optimization toolbox provides an industrial strength solver capable of solving huge problems that other less specialized MATLAB packages can't solve.

2.2.1 Features of the MOSEK optimization toolbox

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

- Solve linear optimization problems.

---

[1] MATLAB is made by MathWorks, see http://www.mathworks.com.

  - Using either an interior-point or a simplex optimizer.

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

The items in bold above is new possibilities in version of the MOSEK optimization toolbox for MATLAB.

2.3 Comparison with the MATLAB optimization toolbox

MathWorks the maker of MATLAB also sells an optimization toolbox so an obvious question is how these two products compares on the following issues[2]:

Problem types: In general the MOSEK optimization toolbox can only solve **convex** optimization problems whereas the MATLAB toolbox also handles nonconvex problems.

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

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

The main computational engine within the MOSEK optimization toolbox is a primal-dual type interior-point algorithm which has been demonstrated to be very well-suited for solution of large-scale problems. Particularly when the algorithm is implemented

---

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

using state-of-the-art (sparse) linear algebra as is the case for the MOSEK optimization toolbox. Readers interested in the details are referred to [5].

However, it should be mentioned a primal simplex optimizer is available for linear problems.

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

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

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

The only differences between the functionality of the MOSEK and MATLAB version of these functions are that the MOSEK version does not use all the MATLAB options, does not use an optional starting point[3], and the MOSEK version of `quadprog` is only intended for convex problems. On the other hand then the large-scale version of the MATLAB optimization toolbox does not accept arbitrary bounds and linear side constraints for quadratic problems whereas MOSEK does.

In general for problems that both the MATLAB optimization toolbox and MOSEK optimization toolbox solves, then MOSEK delivers the best reliability and performance but of course MOSEK cannot solve many problems the MATLAB optimization toolbox deals with.

---

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

Chapter 3

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.

http://www.mosek.com/fileadmin/products/5%5f0/tools/doc/html/toolsinstall/node003.html

3.1 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 shows how to use `addpath`.

3.1.1 On Windows

```
% If you use MATLAB 7.4 (R2007a) or any later version do
addpath 'c:\Program Files\mosek\5\toolbox\r2007a'

% For  MATLAB 7.3 (R2006b)
addpath 'c:\Program Files\mosek\5\toolbox\r2006b'

% For  MATLAB 7.2 (R2006a)
addpath 'c:\Program Files\mosek\5\toolbox\r2006a'
```

This assumes you installed MOSEK at

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

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

3.1.2 On Linux/UNIX/MAC OSX

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

```
% For MATLAB 7.4 (R2007a) or any later version  do
addpath '/home/user/mosek/5/toolbox/r2007a'

% For MATLAB 7.3 (R2006b)
addpath '/home/user/mosek/5/toolbox/r2006b'

% For MATLAB 7.2 (R2006a)
addpath '/home/user/mosek/5/toolbox/r2006a'
```

This assumes MOSEK is installed at

```
/home/user
```

3.1.3 Permanently changing `matlabpath`

Normally you will have to do `addpath` command every time MATLAB is started. However, it can be avoided if the addpath command is added to the file

```
<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 menu item

```
\File\Set Path
```

3.2 Verifying MOSEK works

You can verify that MOSEK works by executing the command

```
mosekopt
```

inside MATLAB. In case MOSEK you should get something like

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


    MOSEK command summary.

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

If you do not get that, then please read Section 3.3.

3.3 Troubleshooting

3.3.1 ??? Undefined function or variable 'mosekopt'

In the case you get the MATLAB error message

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

you have not setup the `matlabpath` correctly as described in Section 3.1.

3.3.1.1 Unable to load mex file

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

The other possible reason is discussed below.

- Windows:

  MATLAB reports something like

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

  This problem is most likely caused by MOSEK cannot load the MOSEK DLL which in turn is caused by the operating system variable

  ```
  PATH
  ```

  is not appropriately setup.

  Please consult the "MOSEK optimization tools installation manual" and read about how to install MOSEK under Windows and how to setup the operating system variable `PATH`.

- MAC OSX:

  The problem is that operating system variable `DYLD_LIBRARY_PATH` variable is not appropriately setup. Setting this variable can be tricky. In particularly if you are invoking MATLAB by clicking on the MATLAB icon. In this case a file named

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

  with a proper content should exists on your computer. Further details about the file `environment.plist` and how to install MOSEK under MAC OSX can be seen in the "MOSEK optimization tools installation manual".

- UNIX:

  MATLAB reports something like

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

  The cause of the problem is that the shared library

  ```
  libmosek.so.2.5
  ```

  cannot be loaded. This problem normally is caused by that the OS environment variable

  ```
  LD_LIBRARY_PATH
  ```

  is not appropriately setup. Observe that `LD_LIBRARY_PATH` may have another name on some UNIX systems. Please consult the "MOSEK optimization tools installation manual" and read about how to install MOSEK under UNIX.

3.3.2 `libgcc_s.so.1 must be installed for pthread_cancel to work`

This error is caused by an old version of the library

```
libgcc_s.so.1
```

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

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

before running MATLAB.

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

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

to

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

and the problem should be solved.

3.3.3 Compiling with the MATLAB compiler

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

3.3.4 Shadows the M-file

If you encounter the error

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

when compiling a MATLAB script using MOSEK then you must delete the file

```
c:\mosek\5\toolbox\<MATLABVERSION>\mosekopt.m
```

This should fix the compile error.

3.3.5 Cannot find authentication file

If you encounter the error

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

??? Invalid MEX-file 'C:\mosek\4\toolbox\r2006b\mosekopt.mexw32': .
```

The try removing any `addpath` commands from your code when compiling. Instead, specify the location of the MOSEK files with

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

in the compile command.

Chapter 4

Getting support and help

4.1 MOSEK documentation

For an overview of the available MOSEK documentation please see

`mosek\5\help\index.html`

in the distribution.

4.2 Additional reading

In this manual it is assumed the reader is familiar with mathematics and in particular mathematical optimization. Some introduction to linear programming can be found in books such as “Linear programming” by Chvátal [17] or “Computer Solution of Linear Programs” by Nazareth [22]. For more theoretical aspects see for example “Nonlinear programming: Theory and algorithms” by Bazaraa, Shetty, and Sherali [12]. Finally the book “Model building in mathematical programming” by Williams [27] provides an excellent introduction to modelling issues in optimization.

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

Chapter 5

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 MOSEK optimization toolbox for MATLAB. The novice user can safely skip to the next chapter.

5.1 MOSEK replacements for MATLAB functions

MOSEK provides replacements for the MATLAB functions:

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

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

5.2 The license system

By default MOSEK caches some information about the license from each call of `mosekopt` to the next. This greatly speed up license checkout. License caching can be disabled with the command `'nokeepenv'` to `mosekopt`.

Chapter 6

A guided tour

6.1 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 and read big manuals. 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 guide tour to the optimization toolbox has been designed which introduces the toolbox using examples. The intention is that after studying these examples, then the reader should be able to solve his or her own optimization problems without much further effort. Nevertheless, for the user who is interested in exploiting the toolbox to the limits, then a detailed discussion and command reference is provided in the subsequent chapters.

6.2 The tour starts

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

`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 MOSEK optimization toolbox consists of one procedure named `mosekopt`. This procedure provide 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 then MOSEK also provides an implementation of `linprog`, `quadprog` and so forth. For details about these functions we refer the reader to Chapter 7.

In the subsequent sections the use of MOSEK optimization toolbox is demonstrated using examples. Most of these examples are available in the directory

```
mosek\5\toolbox\examp\
```

6.3 The MOSEK terminolgy

First some MOSEK terminology is introduced which will make the subsequent 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 are 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. For example the interior-point optimizers produces a general interior-point solution whereas the simplex optimizer produces a basic solution.

6.4 Linear optimization

The first example is the linear optimization problem

$$\begin{array}{lccccc} \text{minimize} & & & x_1 + 2x_2 & & \\ \text{subject to} & 4 & \leq & x_1 + x_3 & \leq & 6, \\ & 1 & \leq & x_1 + x_2, & & \\ & & & 0 \leq x_1, x_2, x_3. & & \end{array} \tag{6.1}$$

6.4.1 Using `msklpopt`

A linear optimization problem such as (6.1) can be solved using the `msklpopt` function which is designed for solution of the problem

$$\begin{array}{lccccc} \text{minimize} & & & c^T x & & \\ \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x. \end{array} \tag{6.2}$$

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

The first step of solving the example (6.1) is to setup the data for problem (6.2) i.e. the c, A, etc. Afterwards the problem is solved using an appropriate call to `msklpopt`.

```
% lo1.m

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

% Ineterior-point solution.

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

% Basic solution

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

Note that

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

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

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

6.4.2 Using `mosekopt`

The function `msklpopt` is in fact just a wrapper around the real optimization routine `mosekopt`. Therefore, an alternative to use the `msklpopt` is to call `mosekopt` directly if desired. 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`	Is a string telling what `mosekopt` should do. For example `'minimize info'` tells `mosekopt` that the objective should be minimized and information about the optimization should be returned.
`prob`	A MATLAB structure specifying the problem that should be optimized.
`param`	A MATLAB structure specifying parameters controlling the behaviour of the MOSEK optimizer. However, in general it should not be necessary to change the parameters.

The following MATLAB commands demonstrate how to setup the `prob` structure for the example (6.1) and solve the problem using `mosekopt`:

```
% lo2.m

clear prob;

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

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

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

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

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

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

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

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

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

Observe 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`.

6.5 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:

$$\begin{array}{lccc} \text{minimize} & & & x_1^2 + 0.1x_2^2 + x_3^2 - x_1x_3 - x_2 \\ \text{subject to} & 1 & \leq & x_1 + x_2 + x_3 \\ & & & x \geq 0. \end{array} \tag{6.3}$$

In general a quadratic optimization problem has the form

$$\begin{array}{lccccc} \text{minimize} & & & \frac{1}{2}x^TQx + c^Tx & & \\ \text{subject to} & l^c & \leq & Ax, & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \end{array} \tag{6.4}$$

which for the example (6.3) implies

$$Q = \begin{bmatrix} 2 & 0 & -1 \\ 0 & 0.2 & 0 \\ -1 & 0 & 2 \end{bmatrix}, \quad c = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, \tag{6.5}$$

and

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

Note the explicit $\frac{1}{2}$ in the objective function of (6.3) which implies diagonal elements must be doubled in Q i.e. $Q_{11} = 2$, whereas the coefficient in (6.4) is 1 in front of x_1^2.

6.5.1 Two important assumptions

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

$$Q = Q^T$$

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

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

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

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

6.5.2 Using `mskqpopt`

The subsequent MATLAB statements solve the problem (6.3) using the MOSEK function `mskqpopt`

```
% qo1.m

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

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

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

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

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

6.5.3 Using `mosekopt`

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

```
% qo2.m

clear prob;

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

% Defining the data.

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

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

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

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

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

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

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

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

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

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

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

% Display primal solution for the variables
```

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

This sequence of commands looks much like the one that was used to solve the linear optimization example using `mosekopt`except for 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

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

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

6.6 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 toolbox function `mosekopt`.

6.6.1 The conic optimization problem

To be specific a conic optimization problem has the following form

$$\begin{array}{lrcccl} \text{minimize} & & & c^T x + c^f & & \\ \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \\ & & & x \in \mathcal{C}, & & \end{array} \tag{6.6}$$

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

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

be vectors comprised of parts of the decision variables x such that each decision variable is a member of exactly **one** vector x^t. For example it could be the case that

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

Next define

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

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

- R set:

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

- Quadratic cone:

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

- Rotated quadratic cone:

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

The R set is never specified explicitly, because if a variable is not a member of any other cone, then it is member of this cone.

Although the cones MOSEK can handle give rise to a limited class of conic problems, then it includes linear, quadratic, quadratically constrained optimization, and other classes of nonlinear convex optimization problems. See Section 9.4 for a discussion.

6.6.2 Solving an example

The problem

$$\begin{array}{lccc} \text{minimize} & x_5 + x_6 & & \\ \text{subject to} & x_1 + x_2 + x_3 + x_4 & = & 1, \\ & x_1, x_2, x_3, x_4 & \geq & 0, \\ & x_5 \geq \sqrt{x_1^2 + x_3^2}, & & \\ & x_6 \geq \sqrt{x_2^2 + x_4^2} & & \end{array} \tag{6.7}$$

is an example of a conic quadratic optimization problem. The problem involves some linear constraints and two quadratic cones. The linear constraints are specified just as if the problem where a linear problem whereas the cones are specified using a MATLAB cell array[1] named `cones`. `cones` must contain one cell per cone, where a cell must contain the two fields `type` and `sub`. `type` is used to specify the the type of the cone and `sub` is used to specify the member variables of the cone.

The following MATLAB code demonstrates how to solve the example (6.7) using MOSEK.

---

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

```
% cqo1.m

clear prob;

% First the non conic part of the problem is specified.

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

% Next the cones are specified.

% First an empty cell array named
% cones is defined. It should contain
% one cell per cone.

prob.cones   = cell(2,1);

% The first cone is specified.

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

% The subfield type specifies the cone type
% i.e. whether it is quadratic cone
% or rotated quadratic cone. The keys
% for the two cone types are MSK_CT_QUAD
% MSK_CT_RQUAD respectively.
%
% The subfield sub specifies the members
% of the cone. I.e. the above definition
% implies x(5) >= sqrt(x(3)^2+x(1)^2)

% The second cone is specified.

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

% Finally, the problem is optimized.

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

% The primal solution is displayed.

res.sol.itr.xx'
```

A couple of important comments are:

- No variable must be member of more than one cone. This is not serious restriction. See the subsequent section.
- The R set is not specified explicitly.

6.6.3 Quadratic and conic optimization

The example

$$\begin{array}{lllr} \text{minimize} & x_1 + x_2 + x_3 & & \\ \text{subject to} & x_1^2 + x_2^2 + x_3^2 & \leq & 1, \\ & x_1 + 0.5x_2^2 + x_3 & \leq & 0.5 \end{array} \tag{6.8}$$

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

Indeed the first constraint is equivalent to

$$\begin{array}{lll} x_4 & \geq & \sqrt{x_1^2 + x_2^2 + x_3^2}, \\ x_4 & = & 1 \end{array} \tag{6.9}$$

where x_4 is a new variable. This is quadratic cone and linear constraint. The second constraint in (6.8) 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

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

and

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

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

$$x_2 = x_7$$

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

Therefore, the example (6.8) is equivalent to the conic quadratic optimization problem

$$
\begin{array}{lccl}
\text{minimize} & x_1 + x_2 + x_3 & & \\
\text{subject to} & x_1 + x_3 + x_5 & = & 0.5, \\
 & x_2 - x_7 & = & 0, \\
 & x_4 & = & 1, \\
 & x_5 & \geq & 0, \\
 & x_6 & = & 1, \\
 & \multicolumn{3}{l}{x_4 \geq \sqrt{x_1^2 + x_2^2 + x_3^2},} \\
 & \multicolumn{3}{l}{2x_5x_6 \geq x_7^2.}
\end{array}
\tag{6.10}
$$

This problem can be solved using MOSEK as follows:

```
% cqo2.m

% Setup of the non conic part of the problem.

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

% Setup of cone information.

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

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

% Display the solution.
res.sol.itr.xx’
```

6.6.4 Conic duality and the dual solution

The dual problem corresponding to the conic optimization problem (6.6) is given by

$$\begin{array}{lrcl} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c & & \\ & +(l^x)^T s_l^x - (u^x)^T s_u^x + c^f & & \\ \text{subject to} & -y + s_l^c - s_u^c & = & 0, \\ & A^T y + s_l^x - s_u^x + s_n^x & = & c, \\ & s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0, \\ & s_n^x \in \mathcal{C}^* & & \end{array} \tag{6.11}$$

where the dual cone $\mathcal{C}^*$ is defined as follows. Let (s_n^x) be partitioned similar to x i.e. if x_j is 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

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

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 are given as follows:

- R set:
$$\mathcal{C}_t = \left\{ x \in R^{n^t} \right\} \quad \Leftrightarrow \quad \mathcal{C}_t^* := \left\{ s \in R^{n^t} : \ s = 0 \right\}.$$
- Quadratic cone:
$$\mathcal{C}_t := \left\{ x \in R^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\} \quad \Leftrightarrow \quad \mathcal{C}_t^* = \mathcal{C}_t.$$
- Rotated quadratic cone:
$$\mathcal{C}_t := \left\{ x \in R^{n^t} : 2x_1x_2 \geq \sqrt{\sum_{j=3}^{n^t} x_j^2}, \ x_1, x_2 \geq 0 \right\}. \quad \Leftrightarrow \quad \mathcal{C}_t^* = \mathcal{C}_t.$$

For a more detailed discussion about conic duality see Section 9.4.

6.6.4.1 How to obtain the dual solution

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

```
% cqo3.m

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

```
% Solution record.
res.sol

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

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

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

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

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

6.6.5 Setting accuracy parameters for the conic optimizer

Three parameters controls the accuracy of the solution obtained by 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);
```

6.7 Quadratically constrained optimization

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

$$\begin{array}{lll} \text{minimize} & x_1 + x_2 + x_3 & \\ \text{subject to} & x_1^2 + x_2^2 + x_3^2 & \leq \; 1, \\ & x_1 + 0.1x_2^2 + x_3 & \leq \; 0.5. \end{array} \tag{6.12}$$

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

```
% qco1.m

clear prob;

% Specifying problem data.

% First c.
prob.c      = ones(3,1);

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

% a
prob.a      = [sparse(1,3);sparse([1 0 1])];

prob.buc    = [1 0.5]';

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

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

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

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

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

6.8 Linear least squares and related norm minimization problems

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

$$\text{minimize} \quad \|Fx - b\| \tag{6.13}$$

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

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

6.8.1 The case of the 2 norm

However, initially let us focus on the 2 norm. In this case (6.13) is identical to the quadratic optimization problem

$$\text{minimize} \quad 1/2x^TF^TFx + 1/2b^Tb - b^TFx \tag{6.14}$$

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

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

Subsequently, it is demonstrated how the quadratic optimization problem (6.14) is solved using `mosekopt`. In the example the problem data is first read from file. Next data for the problem (6.14) is constructed and finally the problem is solved.

```
% nrm1.m

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

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

% The problem is solved by solving
%             minimize 0.5 xf'fx+0.5*b'*b-(f'*b)'*x

% Clearing prob
clear prob;

% Compute the fixed term in the objective.
% It is not really needed.
prob.cfix = 0.5*b'*b
```

```
% Forming c
prob.c = -f'*b;

% Forming q. Note only the lower triangular
% part of f'*f is used.
[prob.qosubi,prob.qosubj,prob.qoval] = find(sparse(tril(f'*f)))

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

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

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

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

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

Quite frequently the x variables must be within some bounds or satisfy some additional linear constraints. These requirements can easily be incorporated into the problem (6.14). For example the constraint $\|x\|_\infty \leq 1$ can be modeled as follows

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

% Now assume the same objective should be
% minimized subject to -1 <= x <= 1

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

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

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

6.8.2 The case of the infinity norm

In some applications of the norm minimization problem (6.13) it is better to use the infinity norm than the 2 norm. However, the problem (6.13) stated as an infinity norm problem is equivalent to the linear optimization problem

$$\begin{array}{lrcl} \text{minimize} & & \tau & \\ \text{subject to} & Fx + \tau e - b & \geq & 0, \\ & Fx - \tau e - b & \leq & 0, \end{array} \tag{6.15}$$

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

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

and hence at optimum

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

holds.

The problem (6.15) is straightforward to solve.

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

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

clear prob;

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

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

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

6.8.3 The case of the one norm

Finally, in the case of the one norm then by definition we have that

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

Therefore, the norm minimization problem can be formulated as follows

$$\begin{array}{lrcl} \mbox{minimize} & \sum_{i=1}^{m} t_i & & \\ \mbox{subject to} & |f_{i:}x - b_i| & = & t_i, \quad i = 1, \ldots, m, \end{array} \tag{6.16}$$

which in turn is equivalent to

$$\begin{array}{lrcl} \mbox{minimize} & \sum_{i=1}^{m} t_i & & \\ \mbox{subject to} & f_{i:}x - b_i & \leq & t_i, \quad i = 1, \ldots, m, \\ & -(f_{i:}x - b_i) & \leq & t_i, \quad i = 1, \ldots, m. \end{array} \tag{6.17}$$

The reader should verify that this is really the case.

In matrix notation this problem can be expressed as follows

$$\begin{array}{lrcl} \text{minimize} & e^T t & & \\ \text{subject to} & Fx - te & \leq & b, \\ & Fx + te & \geq & b, \end{array} \tag{6.18}$$

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

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

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

clear prob;

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

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

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

6.8.3.1 A better formulation

It is possible to improve upon the formulation of the problem (6.17). Indeed problem (6.17) is equivalent to

$$\begin{array}{lrcll} \text{minimize} & \sum_{i=1}^{m} t_i & & & \\ \text{subject to} & f_{i:}x - b_i - t_i + v_i & = & 0, & i = 1, \ldots, m, \\ & -(f_{i:}x - b_i) - t_i & \leq & 0, & i = 1, \ldots, m, \\ & v_i \geq 0, & & & i = 1, \ldots, m. \end{array} \tag{6.19}$$

After eliminating the t variables then this problem is equivalent to

$$\begin{array}{lrcll} \text{minimize} & \sum_{i=1}^{m} (f_{i:}x - b_i + v_i) & & & \\ \text{subject to} & -2(f_{i:}x - b_i) - v_i & \leq & 0, & i = 1, \ldots, m, \\ & v_i \geq 0, & & & i = 1, \ldots, m. \end{array} \tag{6.20}$$

Note this problem only have the half number of general constraints of problem (6.17). I.e. 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. Also note MOSEK only stores the non-zeros in the coefficient matrix of the constraints. This implies that the problem (6.20) is likely to require much less space than the problem (6.19).

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

6.9 More about solving linear least squares problems

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

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

$$\begin{array}{lccl} \text{minimize} & \frac{1}{2}\left\|Fx-f\right\|_2^2 & & \\ \text{subject to} & Ax & = & b, \\ & l^x \leq x \leq u^x, & & \end{array} \tag{6.21}$$

where F and A are matrices and the remaining quantities are vectors. x is the vector of decision variables. The problem (6.21) as stated is a convex quadratic optimization problem and can be solved as such.

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

$$\begin{array}{lccl} \text{minimize} & \frac{1}{2}\left\|z\right\|_2^2 & & \\ \text{subject to} & Ax & = & b, \\ & Fx-z & = & f, \\ & l^x \leq x \leq u^x. & & \end{array} \tag{6.22}$$

Note a number of new constraints and variables have been introduced which of course seem to be disadvantageous but on the other hand the Hessian of the objective in problem (6.22) is much sparser than in problem (6.21). This frequently turns out to be more important for the computational efficiency and therefore the latter formulation is usually the better one.

In the case F has many more rows than columns, then formulation (6.22) is not attractive but the corresponding dual problem is. Using the duality theory outlined in Section 9.5.1 we obtain the dual problem

$$\begin{array}{lccl} \text{maximize} & b^T y + f^T \bar{y} & & \\ & +(l^x)^T s_l^x + (u^x)^T s_u^x & & \\ & -\frac{1}{2}\left\|z\right\|_2^2 & & \\ \text{subject to} & A^T y + F^T \bar{y} + s_l^x - s_u^x & = & 0, \\ & z - \bar{y} & = & 0, \\ & s_l^x, s_u^x \geq 0 & & \end{array} \tag{6.23}$$

which can be simplified to

$$\begin{array}{lrcl} \text{maximize} & b^T y + f^T z & & \\ & +(l^x)^T s_l^x + (u^x)^T s_u^x & & \\ & -\frac{1}{2} \|z\|_2^2 & & \\ \text{subject to} & A^T y + F^T z + s_l^x - s_u^x & = & 0, \\ & s_l^x, s_u^x \geq 0 & & \end{array} \tag{6.24}$$

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

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

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

Given our assumptions then the dual problem (6.24) will have much fewer constraints than the primal problem (6.22). This is important because MOSEK tends to be more efficient the fewer constraints there are in the problem. One obvious question is what if the dual problem (6.24) is solved instead of the primal problem (6.22), then how is the optimal x solution obtained. It turns that the dual variables corresponding to the constraint

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

is the optimal x solution. Therefore, due to MOSEK always reports this information as the

`res.sol.itr.y`

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

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

```
% nrm5.m

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

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

% In this case there are no linear constraints
% First we solve the primal problem:
%
% minimize 0.5|| z ||^2
% subject  F x - z = f
%          l <= x <= u
% Note m>>n
[m,n]       = size(F);

prob        = [];

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


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

fprintf('First try\n');

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

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

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

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

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

% Clearing prob again.
prob=[];

%
% Next we  solve the dual problem

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

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

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

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

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

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

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

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

Here is the output produced:

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

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

It can be observed that 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. So in this case we can conclude that the dual formulation is far superior to the primal formulation of the problem.

6.9.1 Using conic optimization linear least squares problems

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

$$\begin{array}{lccl} \text{minimize} & \|Fx-f\|_2 & & \\ \text{subject to} & Ax & = & b, \\ & l^x \leq x \leq u^x, & & \end{array} \tag{6.25}$$

is equivalent to

$$\begin{array}{lccccl} \text{minimize} & & t & & \\ \text{subject to} & & Ax & = & b, \\ & & Fx-z & = & f, \\ & l^x \leq & x & \leq & u^x, \\ & & \|z\|_2 \leq t. & & \end{array} \tag{6.26}$$

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

$$\begin{array}{lccl} \text{maximize} & b^T y + f^T \bar{y} + (l^x)^T s_l^x - (u^x)^T s_u^x & & \\ \text{subject to} & A^T y + F^T \bar{y} + s_l^x - s_u^x & = & 0, \\ & -\bar{y} + s_z & = & 0, \\ & s_t & = & 1, \\ & \|s_z\| \leq s_t. & & \\ & s_l^x, s_u^x \geq 0 & & \end{array} \tag{6.27}$$

which can be reduced to

$$\begin{array}{lccl} \text{maximize} & b^T y + f^T s_z + (l^x)^T s_l^x - (u^x)^T s_u^x & & \\ \text{subject to} & A^T y - F^T \bar{s}_z + s_l^x - s_u^x & = & 0, \\ & s_t & = & 1, \\ & \|s_z\| \leq s_t, & & \\ & s_l^x, s_u^x \geq 0. & & \end{array} \tag{6.28}$$

Quite frequently 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 from the dual solution of the dual.

6.10 Entropy optimization

6.10.1 Using `mskenopt`

An entropy optimization problem has the following form

$$\begin{array}{lccccc} \text{minimize} & & \sum_{j=1}^{n} d_j x_j \ln(x_j) + c^T x & & \\ \text{subject to} & l^c \leq & Ax & \leq & u^c, \\ & & 0 \leq x, & & \end{array} \tag{6.29}$$

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

$$\begin{array}{lccccc} \text{minimize} & & x_1 \ln(x_1) - x_1 + x_2 \ln(x_2) & & \\ \text{subject to} & 1 \leq & x_1 + x_2 & \leq & 1, \\ & & 0 \leq x_1, x_2 & & \end{array} \tag{6.30}$$

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

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

6.11 Geometric optimization

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

$$\begin{array}{lccc} \text{minimize} & \sum_{k \in J_0} c_k \prod_{j=1}^{n} t_j^{a_{kj}} & & \\ \text{subject to} & \sum_{k \in J_i} c_k \prod_{j=1}^{n} t_j^{a_{kj}} & \leq & 1, \quad i = 1, \ldots, m, \\ & t > 0, & & \end{array} \tag{6.31}$$

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 an $T \times n$ matrix and c is a vector of length t. In general the problem (6.31) is very hard to solve, but the posynomial case where

$$c > 0$$

is relatively easy. The problem (6.31) is in general not a convex optimization problem, but using the variable transformation

$$t_j = e^{x_j} \tag{6.32}$$

we obtain the problem

$$\begin{array}{lll} \text{minimize} & \sum_{k \in J_0} c_k e^{a_{k:} x} & \\ \text{subject to} & \sum_{k \in J_i} c_k e^{a_{k:} x} \leq 1, & i = 1, \ldots, m, \end{array} \tag{6.33}$$

Now using that the log function is an increasing function we obtain an equivalent problem

$$\begin{array}{lll} \text{minimize} & \log(\sum_{k \in J_0} c_k e^{a_{k:} x}) & \\ \text{subject to} & \log(\sum_{k \in J_i} c_k e^{a_{k:} x}) \leq \log(1), & i = 1, \ldots, m, \end{array} \tag{6.34}$$

which is a convex optimization problem. Hence, the problem (6.34) can be solved by MOSEK.

For further details about geometric optimization we refer the reader to [12, pp. 531-538].

6.11.1 Using `mskgpopt`

MOSEK cannot solve a geometric optimization problem directly. However, after it has been transformed to the form (6.34), then it can be solved using the MOSEK optimization toolbox function `mskgpopt`. Note that the solution to the transformed problem can easily be converted into a solution to the original geometric optimization problem using relation (6.32).

Subsequently, we will use the example

$$\begin{array}{lll} \text{minimize} & 40 t_1^{-1} t_2^{-1/2} t_3^{-1} + 20 t_1 t_3 + 40 t_1 t_2 t_3 & \\ \text{subject to} & \frac{1}{3} t_1^{-2} t_2^{-2} + \frac{4}{3} t_2^{1/2} t_3^{-1} & \leq 1, \\ & 0 < t_1, t_2, t_3 & \end{array} \tag{6.35}$$

to demonstrate how a geometric optimization problem is solved using `mskgpopt`. Note that both the objective and the constraint functions consists of a sum of similar type of terms. These terms and where they belong can be specified completely using the matrix

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

and the vectors

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

The interpretation is that each row of A and c describe 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 tells whether a term belongs to the objective or a constraint. If map_k is equal to zero, then the kth term belongs to the objective function. Otherwise it belongs to the map_kth constraint.

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

```
% go1.m

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

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

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

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

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

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

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

6.11.2 Comments

6.11.2.1 Solving large scale problems

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

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

where

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

Please try

```
help sparse
```

inside MATLAB for more details about the `sparse` function.

6.11.2.2 Preprocessing tip

Before solving a geometric optimization problem then it is worthwhile to check if a column of the A matrix inputted to `mskgpopt` contains only positive elements. It is easy to verify that if this is the case then the corresponding variable t_i can take the value zero in the optimal solution. This may cause MOSEK problems and it is better to remove such variables from the problem.

6.12 Separable convex optimization

In this section we will discuss solution of nonlinear **separable** convex optimization problems. A general separable nonlinear optimization problem can be specified as follows:

$$\begin{array}{lrcccl} \text{minimize} & & & f(x) + c^T x & & \\ \text{subject to} & & & g(x) + Ax - x^c & = & 0, \\ & l^c & \leq & x^c & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \end{array} \tag{6.36}$$

where

- m is the number of constraints.
- n is the number of decision variables.
- $x \in R^n$ is a vector of decision variables.

- $x^c \in R^m$ is a vector of constraint or slack variables.
- $c \in R^n$ is the part linear objective function.
- $A \in R^{m \times n}$ is the constraint matrix.
- $l^c \in R^m$ is the lower limit on the activity for the constraints.
- $u^c \in R^m$ is the upper limit on the activity for the constraints.
- $l^x \in R^n$ is the lower limit on the activity for the variables.
- $u^x \in R^n$ is the upper limit on the activity for the variables.
- $f : R^n \rightarrow R$ is a nonlinear function.
- $g : R^n \rightarrow R^m$ is a nonlinear vector function.

This implies that the ith 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 (6.36) must satisfy the three important requirements:

1. Separability: This requirement implies that all nonlinear functions can be written on the form
$$f(x) = \sum_{j=1}^{n} f^j(x_j)$$
and
$$g_i(x) = \sum_{j=1}^{n} g_i^j(x_j).$$
Hence, the nonlinear functions can be written as a sum of functions which only depends on one variable.

2. Differentiability: All functions be twice differentiable for all x_j satisfying
$$l_j^x < x < u_j^x$$
if x_j occur in at least one nonlinear function. Hence, if $\sqrt{x_2}$ appears in the problem, then the lower bound on x_2 should be 0.

3. Convexity: The problem should be a convex optimization problem. See Section 9.5 for a discussion of this requirement.

6.12.1 Using `mskscopt`

Subsequently, we will use the following example

$$\begin{array}{lll} \text{minimize} & x_1 - \ln(x_1 + 2x_2) & \\ \text{subject to} & x_1^2 + x_2^2 & \leq 1 \end{array} \tag{6.37}$$

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

First observe the problem (6.37) is not a separable optimization problem due to the logarithmic term in objective is not a function of a single variable. However, by introducing one additional constraint and variable then the problem can be made separable as follows

$$\begin{array}{lrll} \text{minimize} & x_1 - \ln(x_3) & & \\ \text{subject to} & x_1^2 + x_2^2 & \leq & 1, \\ & x_1 + 2x_2 - x_3 & = & 0, \\ & x_3 \geq 0. & & \end{array} \tag{6.38}$$

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

$$x_3 > 0.$$

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

When using the `mskscopt` function to solve problem (6.38), then the linear part of the problem such as a c and A are specified as usual using MATLAB vectors and matrices. However, the nonlinear functions must be specified using five arrays which in the case of problem (6.38) 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,:)` specify the type of a nonlinear function, `opri(k)` specify in which constraint the nonlinear function should be added to (zero means objective), and `oprj(k)` means the nonlinear function should be taken of variable 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)	$f x_j \ln(x_j)$
`exp`	i	j	f	g	$f e^{g x_j}$
`log`	i	j	f	(not used)	$f \ln(x_j)$
`pow`	i	j	f	g	$f x_j^g$

The i value indicates which constraint the nonlinear function belongs too. 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 nonlinear convex separable optimization problem has now been discussed and therefore we will conclude this section by showing the MATLAB code that will solve the example problem (6.38).

```
% sco1.m

% First the linear part of the problem
% is specified.

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

% Then the nonlinear part.

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

% Finally, the optimizer is called. Note that bux is an optional
% parameter which should be added if the variables has an upper
% bound.

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


% Then the solution is printed.
res.sol.itr.xx
```

6.13 Mixed integer optimization

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

MOSEK is capable of solving linear and quadratic optimization problems where one or more of the variables are integer constrained using a mixed integer optimizer[2]

6.13.1 Solving an example

Using the example

$$\begin{array}{lccl} \text{minimize} & -2x_1 - 3x_2 & & \\ \text{subject to} & 195x_1 + 273x_2 & \leq & 1365, \\ & 4x_1 + 40x_2 & \leq & 140, \\ & x_1 \leq 4, & & \\ & x_1, x_2 \geq 0, & & \text{and integer} \end{array} \tag{6.39}$$

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

```
% milo1.m

% First specify the linear problem data as if
% the problem is a linear optimization
% problem.

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

% Specifies indexes of variables that are integer
% constrained.

prob.ints.sub = [1 2];

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

try
  % Display the optimal solution.
```

---

[2] The mixed integer optimizer is a separately licensed option.

```
  res.sol.int
  res.sol.int.xx'
catch
  fprintf('MSKERROR: Could not get solution')
end
```

Observe that compared to a linear optimization problem with no integer constrained variables then:

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

6.13.2 Speeding up the solution of a mixed integer problem

In general a mixed integer optimization problem can be very difficult to solve. Therefore, in some case 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 [28]. However, two methods for assisting the mixed integer optimizer are discussed subsequently.

6.13.2.1 Specifying an initial feasible solution

In many cases a good feasible integer solution may be known to the optimization problem. If that is the case, then it is worthwhile to inform the mixed integer optimizer about it. The reason is that this reduces the space in which the optimizer has to look for an optimal solution.

Consider the problem:

$$\begin{array}{lccl} \text{maximize} & 7x_0 + 10x_1 + x_2 + 5x_3 & & \\ \text{subject to} & x_0 + x_1 + x_2 + x_3 & \leq & 2.5 \\ & x_3 \geq 0 & & \\ & x_0, x_1, x_2 \geq 0 & & \text{and integer,} \end{array} \tag{6.40}$$

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

$$x_0 = 0, x_1 = 2, x_2 = 0, x_3 = 0.5 \tag{6.41}$$

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

```
% milo2.m

clear prob
clear param
```

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


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

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

%Optionally also set status keys
%prob.sol.int.skx = [sc.MSK_SK_SUPBAS;sc.MSK_SK_SUPBAS;...
%                     sc.MSK_SK_SUPBAS;sc.MSK_SK_BAS]
%prob.sol.int.skc = [sc.MSK_SK_UPR]

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

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

It is also possible to specify only the values of the integer variables and then let MOSEK computes values for the remaining continuous variables such that a feasible solution is obtained. If the parameter `MSK_IPAR_MIO_CONSTRUCT_SOL` is set to `MSK_ON` then MOSEK try 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 only necessary to specify 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 (6.40) using a feasible starting solution generated from the integer values as $x_0 = 0, x_1 = 2, x_2 = 0$.

```
% milo3.m

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

clear prob

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

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

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

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

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

6.13.2.2 Using branching priorities

The mixed integer optimized in MOSEK employs the so-called *branch-and-bound* algorithm to search for the optimal solution. See [28, pp. 91-112] for details about the branch-and-bound algorithm. The branch-and-bound algorithm can benefit from knowing about priorities of the integer variables.

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

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

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

This implies that variable 4 has been assigned priority 5 and so forth.

An example of the usage of priorities can be seen in [28, pp. 232-235].

6.14 Sensitivity analysis

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

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

Consider the problem:
minimize

$$1x_{11} + 2x_{12} + 5x_{23} + 2x_{24} + 1x_{31} + 2x_{33} + 1x_{34} \tag{6.42}$$

subject to

$$\begin{array}{cccccccccccccc}
x_{11} & + & x_{12} & & & & & & & & & & \leq & 400, \\
 & & & x_{23} & + & x_{24} & & & & & & & \leq & 1200, \\
 & & & & & & & x_{31} & + & x_{33} & + & x_{34} & \leq & 1000, \\
x_{11} & & & & & & + & x_{31} & & & & & = & 800, \\
 & & x_{12} & & & & & & & & & & = & 100, \\
 & & & x_{23} & + & & & & & x_{33} & & & = & 500, \\
 & & & & & x_{24} & + & & & & & x_{34} & = & 500, \\
x_{11}, & & x_{12}, & x_{23}, & & x_{24}, & & x_{31}, & & x_{33}, & & x_{34} & \geq & 0.
\end{array} \tag{6.43}$$

The example below demonstrate how sensitivity analysis can answer questions of the type: “What happens to the optimal solution if we decrease the upper bound on the first constraint with 1”. For more information on sensitivity analysis see Chapter 13.

```
% sensitivity2.m

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

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

```
% analyse upper bound on constraint 1
prob.prisen.cons.subu = [1];

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

The output from running the example is given below:

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

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

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

6.15 The solutions

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

```
res.sol
```

which have one or more of the subfields

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

The interior (point) solution is an arbitrary optimal solution which is computed using the interior-point optimizer. The basis solution is only available 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 only available for problems having integer constrained variables and is computed using integer optimizer.

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

`.prosta`	Problem status. See Section D.34.
`.solsta`	Solution status. See Section D.44.
`.skc`	Constraint status keys. See Section 6.1 below.
`.skx`	Variable status keys. See Section 6.1 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.

6.15.1 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 6.1 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

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

Table 6.1: Constraint and variable status keys.

```
  % 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.

6.16 Viewing the task information

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

All the items stored in the task information database are listed in Sections D.15 and D.11. It is of course possible to see the whole or part of information task database 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
```

6.17 Inspecting and setting parameters

A large number of parameters controls the behavior of MOSEK. For example 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 C where all the parameters are presented in details.

In general it should not be necessary to change any of the parameters but if it is required, then it is easy to do so. In the subsequent 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);
```

6.18 Advanced start (warmstart)

In practice it frequently occurs that when an optimization problem has been solved, then the same problem slightly modified should be reoptimized. Moreover, if the modification is only small, then 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. Indeed this is important topic for research. However, the simplex optimizer can exploit any basic solution.

6.18.1 Some examples using warmstart

Using the example

$$\begin{array}{lcccccc} \text{minimize} & & & x_1 + 2x_2 & & \\ \text{subject to} & 4 & \leq & x_1 + x_3 & \leq & 6, \\ & 1 & \leq & x_1 + x_2, & & \\ & & & 0 \leq x_1, x_2, x_3. & & \end{array} \tag{6.44}$$

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

```
% advs1.m

clear prob param bas

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

prob.sol.bas = bas;

% Specify the problem data.
prob.c       = [ 1 2 0]';
subi         = [1 2 2 1];
```

```
subj          = [1 1 2 3];
valij         = [1.0 1.0 1.0 1.0];
prob.a        = sparse(subi,subj,valij);
prob.blc      = [4.0 1.0]';
prob.buc      = [6.0 inf]';
prob.blx      = sparse(3,1);
prob.bux      = [];

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

Some comments are:

- In the example the dual solution is defined. This is acceptable because the primal simplex optimizer is used for the re optimization 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, then it will be important that a “good” dual solution is specified.
- The status keys `bas.skc` and `bas.skx` must only contain the entries `BS`, `EQ`, `LL`, `UL`, and `SB`. Moreover, for example `EQ` must only be specified for a fixed constraint or variable. `LL` and `UL` can only be used for a variable that has a finite lower and 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.

6.18.2 Adding a new variable

Next assume that the problem

$$
\begin{array}{lccccc}
\text{minimize} & & & x_1 + 2x_2 - x_4 & & \\
\text{subject to} & 4 & \leq & x_1 + x_3 + x_4 & \leq & 6, \\
 & 1 & \leq & x_1 + x_2, & & \\
 & & & 0 \leq x_1, x_2, x_3, x_4. & &
\end{array}
\tag{6.45}
$$

should be solved which is identical to the problem (6.44) except a new variable x_4 has been added. In continuation to the previous example this problem can be solved as follows (using warmstart):

```
% advs2.m. Continuation of advs1.m.

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

% Reuse old optimal basis solution.
```

```
bas           = res.sol.bas;

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

% New variable is at the lower limit
bas.xx        = [res.sol.bas.xx;0.0];
bas.slx       = [res.sol.bas.slx;0.0];
bas.sux       = [res.sol.bas.sux;0.0];

prob.sol.bas = bas;

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

% New primal optimal solution
res.sol.bas.xx'
```

6.18.3 Fixing a variable

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

```
% advs3.m. Continuation of advs2.m.

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

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

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

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

The variable x_4 is simply fixed at the value 1 and the problem is re optimized. Note the basis from the previous optimization can immediately be reused.

6.18.4 Adding a new constraint

Now assume that the constraint

$$x_1 + x_2 \geq 2 \tag{6.46}$$

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

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

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

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

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

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

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

res.sol.bas.xx'
```

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

6.18.5 Using numeric values to represent status key codes

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

```
% sk1.m

% Obtain all symbolic constants
% defined in MOSEK.

clear prob bas;

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

% Specify an initial basic solution.
% Note symbolic constants are used.
% I.e. sc.MSK_SK_LOW instead of 4.
```

```
bas.skc      = [sc.MSK_SK_LOW;sc.MSK_SK_LOW];
bas.skx      = [sc.MSK_SK_BAS;sc.MSK_SK_LOW;sc.MSK_SK_BAS];
bas.xc       = [4 1]';
bas.xx       = [1 3 0]';
prob.sol.bas = bas;

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

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

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

% Status keys will be numeric now i.e.

res.sol.bas.skc'

% is vector of numeric values.
```

Note using the command

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

then all the symbolic constants defined within MOSEK are obtained and those constants are then 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 for example MOSEK later change the definition that 1 is equivalent `'LL'`, then all programs using numerical keys are incorrect whereas using the symbolic constants the programs remain correct.

6.19 Using names

In MOSEK it possible to give the objective, each constraint, each variable, and each cone a name. In general there is not much use for such names except in connection with reading and writing MPS files. See Section 6.20 for details.

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

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

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

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

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

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

6.19.1 Blanks in names

Although legal then is strongly advised not to use blanks in names except for the problem name. For instance `'x 1'` should be avoided if possible.

6.20 MPS files

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

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

6.20.1 Reading a MPS file

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

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

In this case `res.prob` will contain several fields which contains the problem data as read from the MPS file. For example the MATLAB command

```
res.prob.c’
```

will display c on the screen.

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

```
% All names.
prob.names

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

It is of course also possible to read problems having quadratic terms in the objective function or the constraints. The following set of MATLAB commands demonstrates how to read such a problem and viewing the data.

```
% mpsrd.m

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

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

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

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

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

6.20.2 Writing a MPS files

It is also possible to write an MPS file using MOSEK. Indeed assume that a problem defined by the MATLAB structure `prob` should be written to a MPS file. This happens in the example:

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

If the field `prob.names` is defined, then MOSEK will use those names when writing the MPS file. Otherwise MOSEK will use generic names.

6.21 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.

6.21.1 Controlling log printing via call-back

When using `mosekopt` it is possible to control the amount of information that `mosekopt` prints to the screen. For instance the command

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

forces `mosekopt` to print no log information because `echo(0)` has been added to the command string. A high number in the command`echo(n)` i.e. for instance `echo(3)` forces MOSEK prints more log information to the screen.

It is possible to redirect the MOSEK log printing almost anywhere using a user defined log call-back function. It works as follows. First create an m-file with a function looking like

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

fprintf(handle,'%s',str);
```

It is not important what the function is called and it is not important what the function does. However, it is important it accepts two input arguments. The first argument is `handle` which is user defined MATLAB structure and the second argument is `str` which is a line of MOSEK log. (Note `myprint` prints the log line to the screen and to a file.)

The following code fragment shows how to inform MOSEK about the function `myprint`.

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

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

6.21.2 The iteration call-back function

It is possible to specify an iteration callback function to be called frequently during the optimization. A typically use for this call-back function is to displays information about the optimization process or to terminate it.

The iteration call-back function takes the following form:

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

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

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

if handle.symbcon. MSK_CALLBACK_INTPNT==where
    % print primal objective
    fprintf('Interior-point primal obj.: %e\n',info.MSK_DINF_INTPNT_PRIMAL_OBJ);

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

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

The function takes three arguments. The first argument `handle` is a user defined MATLAB structure, the second argument `where` indicates from where in the optimization process the callback was invoked and the third argument `info` is a structure containing information about the process. For details about `info` see Section 7.1.7. If the functions return argument which is nonzero, then optimization process is terminated 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);
```

Chapter 7

Command reference

After studying the examples presented in the previous chapter, then it should be possible to use most of facilities in the MOSEK optimization toolbox. Nevertheless a specification of the main data structures employed by MOSEK and a command reference is provided in the present chapter.

7.1 Data structures

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

7.1.1 `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. This structure is used to represent an optimization problem using a number of subfields.

Subfields:

`.names`	Is a MATLAB structure which contain the problem name, the name of the objective, and so forth. See Section7.1.2.
`.qosubi`	i subscript for element q^o_{ij} in Q^o. See (7.6).
`.qosubj`	j subscript for element q^o_{ij} in Q^o. See (7.6).
`.qoval`	Numerical value for element q^o_{ij} in Q^o. See (7.6).
`.qcsubk`	k subscript for element q^p_{ij} in Q^p. See (7.7).
`.qcsubi`	i subscript for element q^p_{ij} in Q^p. See (7.7).
`.qcsubj`	j subscript for element q^p_{ij} in Q^p. See (7.7).

`.qcval`	Numerical value for element q_{ij}^p in Q^p. See (7.7).
`.c`	Liner term in the objective.
`.a`	Is the constraint matrix and it must be a **sparse matrix** having the same number of rows and columns as there are constraints and variables in the problem. This field should always be defined. Even in the case 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 on 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 $-\infty$.
`.buc`	Upper bounds on the constraints. ∞ denotes an infinite upper bound. If the field is not defined or `buc==[]`, then all the upper bounds are assumed to be equal to ∞.
`.blx`	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. ∞ denotes an infinite upper bound. If the field is not defined or `bux==[]`, then all the upper bounds are assumed to be equal to ∞.

`.ints`	A MATLAB structure which has the subfields
	`.sub; % Required.`
	`.pri; % Subfields.`
	`ints.sub` is a one dimensional which contains the indexes of the integer constrained variables. Hence, `ints.sub` is identical to the set $\mathcal{J}$ in (7.5). `ints.pri` is also a one dimensional array of the length as `ints.pri`. The `ints.pri(k)` is the branching priority assigned to the variable `ints.sub(k)`.

`.cones`	A MATLAB cell array which used to define the conic constraint (7.4). See Section 7.1.3 for a details about this structure.
`.sol`	A MATLAB structure which contains a guess for the optimal solution which some of the optimizers in MOSEK may exploit. See Section 7.1.4 for datails about this structure.
`.prisen`	A MATLAB structure which has the subfields: `.cons.subu` Indexes of constraints, where upper bounds are analyzed for sensitivity.

`.cons.subl`	Indexes of constraints, where lower bounds are analyzed for sensitivity.
`.vars.subu`	Indexes of variables, where upper bounds are analyzed for sensitivity.
`.vars.subl`	Indexes of variables, where lower bounds are analyzed for sensitivity.
`.sub`	Index of variables where coefficients are analysed for sensitivity.

Comments:

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

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} q^o_{ij} x_i x_j + c_j x_j \tag{7.1}$$

subject to the functional constraints

$$l^c_k \leq \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} q^p_{ij} x_i x_j + \sum_{j=1}^{n} a_{kj} x_j \leq u^c_k, \; k = 1, \ldots, m, \tag{7.2}$$

the variable bound constraints

$$l^x_j \leq x_j \leq u^x_j, \; j = 1, \ldots, n. \tag{7.3}$$

and the conic constraint

$$x \in \mathcal{C}. \tag{7.4}$$

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

$$x_j \text{ integer constrained for all } j \in \mathcal{J}. \tag{7.5}$$

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

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

$$Q^o = \begin{bmatrix} q^o_{11} & \cdots & q^o_{1n} \\ \vdots & \cdots & \vdots \\ q^0_{n1} & \cdots & q^o_{nn} \end{bmatrix}.$$

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

$$q^o_{ij} = q^o_{ji}$$

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

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

Q^p: The quadratic terms $q^p_{ij} x_i x_j$ in the kth constraint are stored in the matrix Q^p as follows

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

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

$$q^p_{ij} = q^p_{ji}$$

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

A: The constraint matrix A is given by

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

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

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

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

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

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

cones: Specifies the conic constraint. Let

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

be vectors comprised of parts of the decision variables x such that each decision variable is a member of exactly **one** vector x^t. For example we could have

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

Next define

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

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

- R set:

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

- Quadratic cone:

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

- Rotated quadratic cone:

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

All the parameters of the optimization problem are stored using one or more subfields of the `prob` structure using the naming convention in Table 7.1.

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}
`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	$\lvert\mathcal{J}\rvert$	Yes	$\mathcal{J}$
`cones`	MATLAB cell array	k	Yes	$\mathcal{C}$

Table 7.1: The relation between fields and problem parameters.

In Table 7.1 all the parameters is assigned a type where the type `int` means that all the values in this particular field must be integer. Whereas the type `double` means that the values in the field can be any real number. The relationship between Q^o and Q^p and the subfields of the `prob` structure is as follows:

- The quadratic terms in the objective:

$$q^o_{\texttt{qosubi(t)},\texttt{qoval(t)}} = \texttt{qoval(t)}, \ t = 1, 2, \ldots, \texttt{length}(\texttt{qoval}). \tag{7.6}$$

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

- The quadratic terms in the constraints:

$$q^{\texttt{qcsubk(t)}}_{\texttt{qcsubi(t)},\texttt{qcsubj(t)}} = \texttt{qcval}(\texttt{t}),\ t = 1, 2, \ldots, \texttt{length}(\texttt{qcval}). \tag{7.7}$$

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

7.1.2 `names`

This structure is used to store all the names on 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 ith constraint.
`.var`	Is a MATLAB cell array where `names.var{j}` contains the name of the jth constraint.
`.cone`	Is a MATLAB cell array where `names.cone{t}` contains the name of the tth conic constraint.

7.1.3 `cones`

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

The structure contains the subfields:

`.type`	`cones{t}.type` contains the cone type for the tth cone. The `type` subfield can have either value `'MSK_CT_QUAD'` or `'MSK_CT_RQUAD'` which implies the cone is quadratic or rotated quadratic cone respectively.
`.sub`	`cones{t}.sub` is list of variables indexes specifying which variables are member of the cone.

7.1.4 `sol`

Description:

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

Subfields:

`.itr`	Interior (point) solution which is computed by the interior-point optimizer.
`.bas`	Basis solution which computed by the simplex optimizers and basis identification procedure.
`.int`	Integer solution which is computed by the mixed integer optimizer.

Comments

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

`.prosta`	Problem status. See Section D.34.
`.solsta`	Solution status. See Section D.44.
`.skc`	Constraint status keys. See Section 6.1.
`.skx`	Variable status keys. See Section 6.1.
`.skn`	Conic status keys. See Section 6.1.
`.xc`	Constraint activities i.e. $x_c = Ax$ where x is the optimal solution.
`.xx`	Variable activities i.e. the optimal x solution.
`.y`	Identical `sol.slc-sol.suc`.
`.slc`	Dual solution corresponding to the lower constraint bounds.
`.suc`	Dual solution corresponding to the upper constraint bounds.
`.slx`	Dual solution corresponding to the lower variable bounds.
`.sux`	Dual solution corresponding to the upper variable bounds.
`.snx`	Dual solution corresponding to the conic constraint.

The fields `.skn` and `.snx` cannot occur in the `.bas` `.int` solutions. In addition the fields `.y`, `.slc`, `.suc`, `.slx`, and `.sux` cannot occur in the `int` solution because integer problems does not have a well-defined dual and hence cannot have a dual solution.

7.1.5 `prisen`

Description:

Results of primal sensitivity analysis.

Subfields:

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

7.1.6 `duasen`

Description:

Results of dual sensitivity analysis.

Subfields:

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

7.1.7 `info`

`info` is a MATLAB structure containing a subfield for each item in the MOSEK optimization task database. For instance the field `info.MSK_DINF_BI_CPUTIME` specifies the amount of time spend in the basis identification in the last optimization. In Sections D.15 and D.11 are all the items in the task information database shown.

7.1.8 `symbcon`

`symbcon` is a MATLAB structure containing a subfield for each item MOSEK symbolic constant. For instance the field `symbcon.MSK_DINF_BI_CPUTIME` specifies the value of the symbolic constant `MSK_DINF_BI_CPUTIME`. In Section D are all the symbolic constants shown.

7.1.9 `callback`

`callback` is a MATLAB structure containing which contains the subfields (all are optional):

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

`.log` Is the name of a user defined function which must accept two input arguments i.e. for instance

```
function myfunc(handle,str)
```

`handle` will be identical to `callback.handle` when `myfunc` is called and `str` is a line of the log file.

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

`.iter` Is the name of a user defined function which must accept three input arguments i.e. for instance

```
function myfunc(handle,where,info)
```

`handle` will be identical to `callback.iterhandle` when `myfunc` is called and `str` is a line of the log file. `info` is the current information database. See 7.1.7 for further details about `info` data structure.

7.2 An example of a command reference

All functions are documented using the format:

- `somefunction`

  Description:

  The purpose of the function is presented.

  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 number 3 is specified, then 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 about the use of the function is presented.

7.3 Functions provided by the MOSEK optimization toolbox

- `mosekopt`

  Description

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

  Syntax:

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

  Arguments:

- `cmd` — `cmd` is a string containing commands to MOSEK about what it should be doing. For example the string `'minimize info'` means that the objective should be minimized, and information about the optimization process should be returned in `res.info`. The following commands are recognized by `mosekopt`:

  - `aformat` — In the case the problem data are read from a MPS file, then this command controls in which format `prob.a` is returned. (`prob` is the problem structure returned by `mosekopt`.) If `aformat(0)` is used, then `prob.a` is a sparse matrix. In the case `aformat(1)` is used, then the constraint matrix is given by

    ```
    m = % number of constraints
    n = % number of variables
    a = sparse(prob.a.subi,...
               prob.a.subj,...
               prob.a.val,...
               m,n);
    ```

    Hence, `aformat(1)` means `prob.a` is returned in a simple coordinate wise format.

  - `echo()` — Controls how much information is echoed to the screen. Formally, the command is used as

    ```
    echo(level)
    ```

    where `level` is a nonnegative integer. If `level` is identical to 0 nothing is echoed. If `level` is equal to 3, then all messages and errors are echoed to the screen.

  - `read()` — Data is read from a file. If

    ```
    read(name)
    ```

    is used, then data are read from the file `name`. Otherwise the data file name in the MOSEK parameter database is used.

  - `statuskeys()` — The command `statuskeys(0)` means that all the status keys such as the problem status in the solution is reported using string codes. Whereas the command `statuskeys(1)` means that all the status keys are reported using numeric codes.

  - `minimize` — The objective is minimized.

  - `maximize` — The objective is maximized.

`write()`	The problem data is written to a MPS file. In the case `write(name)` is used, then the problem data is written to the file `name`. Otherwise if only `write` is used, then the data is written to the file specified by the MOSEK parameter database.
`param`	When this command is present, then the parameter database is returned in `res.param`.
`info`	When this command is present, then the task information database is returned in `res.info`. This database contains various task specific information. See Section 7.1.7 for details about the `info` data structure.
`symbcon`	When this command is present, then then the data structure `symbcon` is returned in `res.symbcon`. See Section 7.1.8 for details about the `symbcon` data structure.
`nokeepenv`	Delete the MOSEK environment after each run. This can dramaticly increase license checkout overhead and is therefor only intended as a debug feature.

`prob`	(optional) A MATLAB structure containing the problem data. See Section 7.1 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 field must be of a valid type. For example the value of an string parameter must be a string.
`callback`	(optional) A MATLAB structure defining call back data and functions. See Sections 6.21 and 7.1.9 for details.

Returns:

`rcode`	Return code. The interpretation of the value of the return code is listed in Chapter D.
`res`	(optional) Solution obtained by the interior-point algorithm.

`.sol`	The data structure of the type `sol` is discussed in Section 7.1.
`.info`	A MATLAB structure containing the task information database which contains various task

	related information such as about the number of iterations is used to solve the problem. However, this field is only defined if the string `info` is present in the `cmd` command when `mosekopt` is invoked.
`.param`	A MATLAB structure which contain the complete MOSEK parameter database. However, this field is only defined if the command `param` is present in the `cmd` string when `mosekopt` is invoked.
`.prob`	Contains the problem data if the problem data was read from a MPS file.

Examples:

In the follow example it is demonstrated how to list the whole MOSEK parameter database and change the default optimizer to the primal simplex optimizer when the problem is optimized.

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

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

% Modifying a parameter.
param.MSK_IPAR_OPTIMIZER = 3

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

% View the info record.
res.info
```

- `msklpopt`
- `mskqpopt`
- `mskenopt`
- `mskgpopt`
- `mskscopt`

Description

These functions provides an easy to use but less flexible interface than the `mosekopt` function. In fact these procedures is just a wrapper around the `mosekopt` interface and they consists of 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\matlab\solvers
```

Note for example the command

```
help msklpopt
```

will produce some information about how to use `msklpopt`.

Returns:

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

- `mskgpwri`

Description This functions provides an the means of writing gp problem data to a file format compatible with the command line tool `mskexpopt`.

Syntax:

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

Arguments:

`c,a,map` is data accepted by `mskgpopt`. `filename` is the output file name.

Returns:

Nothing.

- `mskgpread`

Description This functions provides an the means of reading gp problem data from a file format compatible with the command line tool `mskexpopt`.

Syntax:

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

Arguments:

`c,a,map` is data accepted by `mskgpopt`. `filename` is the input file name.

Returns:

Data `c,a,map` as accepted by `mskgpopt`.

7.4 MATLAB optimization toolbox compatible functions

The functions presented in this section are provided as a part of both the MOSEK and MATLAB optimization toolboxes. The MOSEK versions are intended to be highly compatible with MATLAB versions and in practice the small differences should not cause any problems.

7.4.1 For linear and quadratic optimization

- `linprog`

Description

Solves the linear optimization problem:

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

Syntax:

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

Arguments:

`f`	The objective function.
`A`	Constraint matrix for the less-than equal inequalities. Use $A = []$ if there are no inequalities.
`b`	Right-hand side for the less-than equal inequalities. Use $b = []$ if there are no inequalities.

`B`	(optional) Constraint matrix for the equalities. Use $B = []$ if there are no equalities.
`c`	(optional) Right-hand side for the equalities. Use $c = []$ if there are no inequalities.
`l`	(optional) Lower bounds for the variables. Please use $-\infty$ to represent infinite lower bounds.
`u`	(optional) Upper bounds for the variables. Please use ∞ to represent infinite lower bounds.
`x0`	(optional) An initial guess for the starting point. This information is ignored by MOSEK.

options	(optional) An optimization options structure. See the function `optionset` for the definition of the optimizations options structure. `linprog` uses the options.
	`.Diagnostics`
	`.Display`
	`.MaxIter`

Returns:

`x`	The optimal x solution.	
`fval`	Optimal objective value i.e. $f^T x$.	
`exitflag`	Is 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 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 the inequalities.
	`.eqlin`	Lagrange multipliers for the equalities.

Examples:

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

- `quadprog`

Description

Solves the quadratic optimization problem:

$$\begin{array}{lrcl} \text{minimize} & \frac{1}{2}x^T H x + f^T x & & \\ \text{subject to} & Ax & \leq & b, \\ & Bx & = & c, \\ & l \leq x \leq u. & & \end{array} \tag{7.8}$$

Syntax:

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

Arguments:

`H`	Hessian of the objective function. H must be a symmetric matrix. Contrary to MATLAB optimization toolbox, then MOSEK only handles the case where H is positive semidefinite. On the hand MOSEK always computes a global optimum i.e. the objective function does have to be strictly convex.
`f`	See (7.8) for the definition.
`A`	Constraint matrix for the less-than equal inequalities. Use $A = []$ if there are no inequalities.
`b`	Right-hand side for the less-than equal inequalities. Use $b = []$ if there are no inequalities.
`B`	(optional) Constraint matrix for the equalities. Use $B = []$ if there are no inequalities.
`c`	(optional) Right-hand side for the equalities. Use $c = []$ if there are no inequalities.
`l`	(optional) Lower bounds for the variables. Please use $-\infty$ to represent infinite lower bounds.
`u`	(optional) Upper bounds for the variables. Please use ∞ to represent infinite lower bounds.
`x0`	(optional) An initial guess for the starting point. This information is ignored by MOSEK.

options	(optional) An optimization options structure. See the function `optimset` for the definition of the optimizations options structure. `quadprog` uses the options.
	`.Diagnostics`
	`.Display`
	`.MaxIter`

Returns:

`x`	x solution.	
`fval`	Optimal objective value i.e. $\frac{1}{2}x^THx + f^Tx$.	
`exitflag`	Is 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 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.

Examples:

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

7.4.2 For linear least squares problems

- lsqlin

Description

Solves the linear linear least squares problem:

$$\begin{array}{lccc} \text{minimize} & \frac{1}{2}\left\|Cx-d\right\|_2^2 & & \\ \text{subject to} & Ax & \leq & b, \\ & Bx & = & c, \\ & l \leq x \leq u. & & \end{array} \tag{7.9}$$

Syntax:

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

Arguments:

`C`	A matrix. See problem (7.9) for the purpose of the argument.

`d`	A vector. See problem (7.9) for the purpose of the argument.
`A`	Constraint matrix for the less-than equal inequalities. Use $A = []$ if there are no inequalities.
`b`	Right-hand side for the less-than equal inequalities. Use $b = []$ if there are no inequalities.
`B`	(optional) Constraint matrix for the equalities. Use $B = []$ if there are no equalities.
`c`	(optional) Right-hand side for the equalities. Use $c = []$ if there are no equalities.
`l`	(optional) Lower bounds for the variables. Please use $-\infty$ to represent infinite lower bounds.
`u`	(optional) Upper bounds for 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.
`x0`	(optional) An initial guess for the starting point. This information is ignored by MOSEK.
options	(optional) An optimization options structure. See the function `optionset` for the definition of the optimizations options structure. `lsqprog` uses the options.
	`.Diagnostics`
	`.Display`
	`.MaxIter`

Returns:

`x`	The optimal x solution.	
`resnorm`	The squared norm of the residuals i.e. $\|Cx - d\|^2$ evaluated at the optimal solution.	
`residual`	Is the residual $Cx - d$.	
`exitflag`	Is 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 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:

Examples:

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

- `lsqnonneg`

Description

Solves the linear least squares problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\left\|Cx-d\right\|_2^2 \\ \text{subject to} & x \geq 0. \end{array} \tag{7.10}$$

Syntax:

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

Arguments:

`C`	See problem (7.10).
`d`	See problem (7.10).
`x0`	(optional) An initial guess for the starting point. This information is ignored by MOSEK.

options	(optional) An optimizations options structure. See the function `optionset` for the definition of the optimizations options structure. `lsqlin` uses the options.
	`.Diagnostics`
	`.Display`
	`.MaxIter`

Returns:

`x`	x solution.
`resnorm`	Squared of the optimal residuals i.e. $$\left\|Cx-d\right\|^2$$ evaluated at the optimal solution.

`residual`	Is the residual $Cx - d$.	
`exitflag`	Is 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:

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

7.4.3 The optimization options

The procedures in the optimization toolbox is in general dependent on some options which for example control the amount of information displayed and the stopping criteria.

In general due to the MOSEK and MATLAB optimization toolboxes employs different algorithms then the toolboxes uses different options. Therefore, the MOSEK optimization toolbox ignores most of the options recognized by the MATLAB toolbox. In the description of the procedure `optimset` it is shown which MATLAB options MOSEK recoginize.

7.4.3.1 Viewing and modifying the optimization options

- `optimget`

Description

Obtains a value of an optimization parameter.

Syntax:

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

Arguments:

`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, then the value `default` is returned.

Returns:

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

Comments:

See the procedure `optimset` for which parameters that can be set.

Examples:

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

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

- `optimset`

Description

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

However, the optimization option structure can be used to modify all the MOSEK parameters. For a discussion of the MOSEK parameters see Section 6.17.

`.Diagnostics`	Used to control how much diagnostic information which is print. It can take the following values: `off` No diagnostic information is printed. `on` Diagnostic information is printed.
`.Display`	Is a string which can take the following values. `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.

Syntax:

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

Arguments:

`arg1`	(optional) Is allowed to be any of the following two things:	
	Any string	Is the same as using no argument.
	A structure	The argument is assumed to be a structure containing options. These options are copied to the return options.
`param1`	(optional) Is a string containing the name of a parameter that should be modified.	
`value1`	(optional) 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:

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

% Modifies the value of parameter
% display in the optimization
% options structure
opt = optionset(opt,’display,’off’);

% Returns default options
opt = optimset(’whatever’)

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

Chapter 8

Case studies

8.1 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 are known with certainty. However, in practice this is seldom the case. For example data may just be roughly estimated, affect 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.

8.1.1 Introductory example

Consider a toy linear optimization problem as follows:

> A company produces two kinds of drugs, DrugI and DrugII, containing a specific active agent A, which is extracted from raw materials which should be purchased on the market. The drug production data are as follows:
>
> | Drug | Selling price, $ per 1000 packs | Content of agent A, g per 1000 packs | Production expenses per 1000 packs | | |
> |---|---|---|---|---|---|
> | | | | manpower, hours | equipment, hours | operational costs, $ |
> | DrugI | 6,200 | 0.500 | 90.0 | 40.0 | 700 |
> | DrugII | 6,900 | 0.600 | 100.0 | 50.0 | 800 |
>
> There are two kinds of raw materials, RawI and RawII, which can be used as sources of the active agent. The related data are as follows:

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

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

Budget, $	Manpower, hours	Equipment, hours	Capacity of raw materials storage, kg
100,000	2,000	800	1,000

> The problem is *to find the production plan which maximizes the profit of the company.*

The problem can be immediately posed as the following linear programming program:

(Drug) :
maximize

$$\begin{array}{ll}
-\overbrace{[100\cdot\texttt{RawI}+199.90\cdot\texttt{RawII}+700\cdot\texttt{DrugI}+800\cdot\texttt{DrugII}]}^{\text{purchasing and operational costs}} & \\
\qquad\qquad +\underbrace{[6200\cdot\texttt{DrugI}+6900\cdot\texttt{DrugII}]}_{\text{income from selling the drugs}} & [\text{total profit}]
\end{array}$$

subject to

$$\begin{array}{rcll}
0.01\cdot\texttt{RawI}+0.02\cdot\texttt{RawII}-0.500\cdot\texttt{DrugI}-0.600\cdot\texttt{DrugII} & \geq & 0 & [\text{balance of active agent}]\\
\texttt{RawI}+\texttt{RawII} & \leq & 1000 & [\text{storage restriction}]\\
90.0\cdot\texttt{DrugI}+100.0\cdot\texttt{DrugII} & \leq & 2000 & [\text{manpower restriction}]\\
40.0\cdot\texttt{DrugI}+50.0\cdot\texttt{DrugII} & \leq & 800 & [\text{equipment restriction}]\\
100.0\cdot\texttt{RawI}+199.90\cdot\texttt{RawII}+700\cdot\texttt{DrugI}+800\cdot\texttt{DrugII} & \leq & 100000 & [\text{budget restriction}]\\
\texttt{RawI},\texttt{RawII},\texttt{DrugI},\texttt{DrugII} & \geq & 0 &
\end{array}$$

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.

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

```
% rlo1.m

clear prob;

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

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

When executing this script, this is what 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 modest, but quite respectful profit 8.8%. Note that at the optimal solution, as it could be guessed in advance, the balance constraint is active: the production process utilizes the full amount of active agent contained in the raw materials.

8.1.2 Data uncertainty and its consequences.

Now note that not all data of the problem could be thought of to be “absolutely reliable”; e.g., one can hardly believe that the contents the active agent in the raw materials are *exactly* the “nominal data” 0.01 g/kg for RawI and 0.02 g/kg for RawII. In reality, these contents definitely vary around the indicated values. A natural assumption here is that the actual contents of active agent a_I in RawI and a_{II} in RawII are realizations of random variables somehow distributed around the “nominal contents” $a_I^{\rm n} = 0.01$ and $a_{II}^{\rm n} = 0.02$. To be more specific, assume that a_I drifts in the 0.5%-margin of $a_I^{\rm n}$, specifically, takes with probabilities 0.5 the values $a_I^{\rm n}(1 \pm 0.005) = \{0.00995; 0.01005\}$. Similarly, assume that a_{II} drifts in the 2% margin of $a_{II}^{\rm n}$, taking with probabilities 0.5 the values $a_{II}^{\rm n}(1 \pm 0.02) = \{0.0196; 0.0204\}$. 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 active agent in raw materials will be less than the one 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 active agent in raw materials is insufficient, the output of the drug is reduced accordingly. With this policy, the actual production of DrugI becomes 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 *pretty small (and unavoidable in reality) 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 can be met in many linear programs of practical origin. Usually, in these programs at least part of the data are not known exactly and can vary around their nominal values, and these data perturbations can make the nominal optimal solution – the one corresponding to the nominal data – infeasible. It turns out that the consequences of data uncertainty can be much more severe than in our toy example. The analysis of linear optimization problems from the NETLIB collection [1] reported in [14] 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, and 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".

8.1.3 Robust linear optimization methodology

A natural approach to handling data uncertainty in optimization is offered by *robust optimization methodology* which, as applied to linear optimization, is as follows.

---

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

8.1.3.1 Uncertain linear programs and their robust counterparts.

Consider a linear optimization problem

$$\begin{array}{llllll} \text{minimize} & & & c^T x & & \\ \text{subject to} & l_c & \leq & Ax & \leq & u_c, \\ & l_x & \leq & x & \leq & u_x, \end{array} \tag{8.1}$$

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 vary in a given *uncertainty set* $\mathcal{U}$. The simplest example is the one of *interval uncertainty*, where every data entry can run through a given interval:

$$\begin{array}{rcl} \mathcal{U} & = & \Big\{ (c, A, l_c, u_c, l_x, u_x) : \\ & & (c^{\rm n} - dc, A^{\rm n} - dA, l_c^{\rm n} - dl_c, u_c^{\rm n} - du_c, l_x^{\rm n} - dl_x, u_x^{\rm n} - du_x) \leq (c, A, l_c, u_c, l_x, u_x) \\ & & \leq (c^{\rm n} + dc, A^{\rm n} + dA, l_c^{\rm n} + dl_c, u_c^{\rm n} + du_c, l_x^{\rm n} + dl_x, u_x^{\rm n} + du_x) \Big\}. \end{array} \tag{8.2}$$

Here $(c^{\rm n}, A^{\rm n}, l_c^{\rm n}, u_c^{\rm n}, l_x^{\rm n}, u_x^{\rm n})$ is the *nominal data*, $(dc, dA, dl_c, du_c, dl_x, du_x) \geq 0$ is the *data perturbation bounds*, and the inequality $\geq$ is understood in the entry-wise sense. Note that some of the entries in the data perturbation bounds can be zero, which means that the corresponding data entries are known exactly – are certain.

- The family of *instances* (8.1) with data running through a given uncertainty set $\mathcal{U}$ is called an *uncertain linear optimization problem.*

- A vector x is called a *robust feasible solution* of an uncertain linear optimization problem, if it remains feasible for all realizations of the data from the uncertainty set, i.e., if

$$l_c \leq Ax \leq u_c,\ l_x \leq x \leq u_x \text{ for all } (c, A, l_c, u_c, l_x, u_x) \in \mathcal{U}. \tag{8.3}$$

- We say that the *robust value of the objective at x* does not exceed a real t, if $c^T x \leq t$ for all realizations of the objective from the uncertainty set.

- 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

$$\min_{t,x} \left\{ t : c^T x \leq t,\ l_c \leq Ax \leq u_c,\ l_x \leq x \leq u_x \text{ for all } (c, A, l_c, u_c, l_x, u_x) \in \mathcal{U} \right\}. \tag{8.4}$$

  The optimal solution to (8.4) is treated as the “uncertainty-immuned” solution to the original uncertain linear programming program.

8.1.3.2 Robust counterpart of uncertain of a linear optimization problem with interval uncertainty.

In general, the RC (8.4) of an uncertain linear optimization problem is not a linear optimization problem (since(8.4) has infinitely many linear constraints). There are, however, cases when (8.4) can be rewritten equivalently as a linear programming program; in particular, this is the case for interval uncertainty (8.2). Specifically, in the case of (8.2), the Robust Counterpart of uncertain linear program is equivalent to the following linear program in variables x, y, t:

$$\begin{array}{lcccccl}
\text{minimize} & & & t & & & \\
\text{subject to} & & & (c^{\mathrm{n}})^T x + (dc)^T y - t & \leq & 0, & (a) \\
 & l_c^{\mathrm{n}} + dl_c & \leq & (A^{\mathrm{n}})x - (dA)y, & & & (b) \\
 & & & (A^{\mathrm{n}})x + (dA)y & \leq & u_c^{\mathrm{n}} - du_c, & (c) \\
 & 0 & \leq & x + y, & & & (d) \\
 & 0 & \leq & -x + y, & & & (e) \\
 & l_x^{\mathrm{n}} + dl_x & \leq & x & \leq & u_x^{\mathrm{n}} - du_x, & (f)
\end{array} \tag{8.5}$$

The origin of (8.5) is quite transparent: the constraints ($8.5.d - e$) 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 (8.5) the entries in the vector $(A^{\mathrm{n}})x - (dA)y$ are lower bounds on the entries of Ax with A from the uncertainty set (8.2), so that ($8.5.b$) ensures that $l_c \leq Ax$ for all data from the uncertainty set. Similarly, ($8.5.c$) and ($8.5.a$), ($8.5.f$) ensure, for all data from the uncertainty set, that that $Ax \leq u_c$, $c^T x \leq t$, and the entries in x satisfy the required lower and upper bounds, respectively.

Note that at the optimal solution to (8.5), 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 necessity to introduce the corresponding additional variable y_i – it can be from the very beginning replaced with x_j, if x_j is nonnegative, or with $-x_j$, if x_j is nonpositive.

Another possible formulation of problem (8.5) is as follows. Let

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

then this equation is equivalent to (a) in (8.5.b). **If $(l_c)_i$ is identical to $-\infty$, then that equation should just be dropped from the computations.** Similarly,

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

is equivalent to (8.5.d). This implies

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

and

$$y = g + x$$

Substituting these values into (8.5) gives

$$
\begin{array}{llcccl}
\text{minimize} & & & t & & \\
\text{subject to} & & & (c^{\mathrm{n}})^T x + (dc)^T (g+x) - t & \leq & 0, \\
& 0 & \leq & f, & & \\
& & & 2(A^{\mathrm{n}})x + (dA)(g+x) + f + l_c^{\mathrm{n}} + dl_c & \leq & u_c^{\mathrm{n}} - du_c, \\
& 0 & \leq & g, & & \\
& 0 & \leq & 2x+g, & & \\
& l_x^{\mathrm{n}} + dl_x & \leq & x & \leq & u_x^{\mathrm{n}} - du_x,
\end{array}
\qquad (8.6)
$$

which after some simplifications leads to

$$
\begin{array}{llccccl}
\text{minimize} & & & t & & & \\
\text{subject to} & & & (c^{\mathrm{n}} + dc)^T x + (dc)^T g - t & \leq & 0, & (a) \\
& 0 & \leq & f, & & & (b) \\
& & & 2(A^{\mathrm{n}} + dA)x + (dA)g + f - (l_c^{\mathrm{n}} + dl_c) & \leq & u_c^{\mathrm{n}} - du_c, & (c) \\
& 0 & \leq & g, & & & (d) \\
& 0 & \leq & 2x+g, & & & (e) \\
& l_x^{\mathrm{n}} + dl_x & \leq & x & \leq & u_x^{\mathrm{n}} - du_x, & (f)
\end{array}
\qquad (8.7)
$$

and

$$
\begin{array}{llccccl}
\text{minimize} & & & t & & & \\
\text{subject to} & & & (c^{\mathrm{n}} + dc)^T x + (dc)^T g - t & \leq & 0, & (a) \\
& & & 2(A^{\mathrm{n}} + dA)x + (dA)g + f & \leq & u_c^{\mathrm{n}} - du_c + l_c^{\mathrm{n}} + dl_c, & (b) \\
& 0 & \leq & 2x+g, & & & (c) \\
& 0 & \leq & f, & & & (d) \\
& 0 & \leq & g, & & & (e) \\
& l_x^{\mathrm{n}} + dl_x & \leq & x & \leq & u_x^{\mathrm{n}} - du_x. & (f)
\end{array}
\qquad (8.8)
$$

Observe this problem has more variables but much fewer constraints than (8.5). Therefore, (8.8) is likely to solver faster than (8.5). Also note (8.8.b) is trivially redundant if $l_x^{\mathrm{n}} + dl_x \geq 0$.

Introductory example (continued). Let us apply the robust optimization methodology to our drug production example presented in Section 8.1.1, assume that the only uncertain data in (Drug) are 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, (Drug) becomes an uncertain LP affected by interval uncertainty; the

Robust Counterpart (8.5) of this uncertain LP is the linear program

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

Solving this problem with MOSEK and 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). 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 adjusting to these data and, with this adjusting, results in the average profit of \$ 7,843, which is by 5.4% *less* than the profit \$ 8,295 *guaranteed* by the robust optimal solution. Note also 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 one. 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 that when possible fluctuations of the contents are taken into the account, RawI turns out to be more profitable than RawII.

8.1.4 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 their "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 look what happens with a particular linear constraint

$$a^T x \leq b \tag{8.9}$$

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

$$a = a^{\mathrm{n}} + \zeta, \tag{8.10}$$

where a^{n} is the vector of nominal coefficients and ζ is a random perturbation vector with zero mean and covariance matrix V_a. In this case, the value of the left hand side of (8.9), 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 which believes that the value of a random variable never exceeds its mean plus 3 times the standard deviation; we do not intend to be that specific and replace "3" in the above rule by a safety parameter Ω which will be in our control. Believing that the value of a random variable "never" exceeds its mean plus Ω times the standard deviation, we conclude that a "safe" version of (8.9) is the inequality

$$(a^{\mathrm{n}})^T x + \Omega\sqrt{x^T V_a x} \leq b. \tag{8.11}$$

The word "safe" above admits a quantitative interpretation: if x satisfies 8.11, then one can bound from above the probability of the event that random perturbations (8.10) result in violating the constraint (8.9) evaluated at x. The bound in question depends on what we know about the distribution of ζ. For example,

1. We always have the bound given by the Tschebyshev inequality:

$$x \text{ satisfies (8.11)} \ \Rightarrow \mathrm{Prob}\left\{a^T x > b\right\} \leq \frac{1}{\Omega^2}. \tag{8.12}$$

2. When ζ is Gaussian, then the Tschebyshev bound can be improved to

$$x \text{ satisfies (8.11)} \ \Rightarrow \mathrm{Prob}\left\{a^T x > b\right\} \leq \frac{1}{\sqrt{2\pi}} \int\limits_{\Omega}^{\infty} \exp\{-t^2/2\} dt \leq 0.5 \exp\{-\Omega^2/2\}. \tag{8.13}$$

3. Assume that $\zeta = D\xi$, where Δ 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 ζ), one has

$$x \text{ satisfies (8.11)} \ \Rightarrow \mathrm{Prob}\left\{a^T x > b\right\} \leq 0.5 \exp\{-\Omega^2/2\}. \tag{8.14}$$

Note that in order to ensure the bounds in (8.13) and (8.14) to be $\leq 10^{-6}$, it suffices to set $\Omega = 5.13$.

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

$$\begin{array}{lrclcl} \text{minimize} & & & [c^{\mathrm{n}} + dc]^T x & & \\ \text{subject to} & (l_c)_i & \leq & [a_i^{\mathrm{n}} + da_i]^T x & \leq & (u_c)_i, \ i = 1, ..., m, \\ & l_x & \leq & x & \leq & u_x, \end{array} \tag{8.15}$$

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 [2]. 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 Ω and replacing the objective and the bodies of all the constraints by their safe bounds as explained above, we arrive at the following optimization problem:

$$\begin{array}{lrclcl} \text{minimize} & & & t & & \\ \text{subject to} & & & [c^{\mathrm{n}}]^T x + \Omega\sqrt{x^T V_c x} & \leq & t, \\ & (l_c)_i & \leq & [a_i^{\mathrm{n}}]^T x - \Omega\sqrt{x^T V_{a_i} x}, & & \\ & & & [a_i^{\mathrm{n}}]^T x + \Omega\sqrt{x^T V_{a_i} x} & \leq & (u_c)_i, \ i = 1, ..., m, \\ & l_x & \leq & x & \leq & u_x. \end{array} \tag{8.16}$$

The relation between problems (8.16) and (8.15) is as follows:

> *If (x, t) is a feasible solution of (8.16), then with probability at least*
>
> $$p = 1 - (m + 1)\exp\{-\Omega^2/2\}$$
>
> *x is feasible for randomly perturbed problem (8.15), and t is an upper bound on the objective of (8.15) evaluated at x.*

We see that if Ω is not too small, (8.16) can be treated as a "safe version" of (8.15).

On the other hand, it is easily seen that (8.16) 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

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

---

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

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* (8.16) *is equivalent to a conic quadratic program*, specifically, to the program

$$
\begin{array}{lcccll}
\text{minimize} & & t & & & \\
\text{subject to} & & [c^{\mathrm{n}}]^T x + \Omega z & \leq & t, & \\
(l_c)_i & \leq & [a_i^{\mathrm{n}}]^T x - \Omega z_i, & & & \\
 & & [a_i^{\mathrm{n}}]^T x + \Omega z_i & \leq & (u_c)_i, \ i = 1, ..., m, & \\
0 & = & w - D_c x & & & \\
0 & = & w^i - D_{a_i} x, & & i = 1, ..., m, & \\
0 & \leq & z - \sqrt{w^T w}, & & & \\
0 & \leq & z_i - \sqrt{(w^i)^T w^i}, & & i = 1, ..., m, & \\
l_x & \leq & x & \leq & u_x. &
\end{array}
\tag{8.17}
$$

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

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

8.1.4.1 Example: Interval and Ellipsoidal Robust counterparts of uncertain linear constraint with independent random perturbations of coefficients.

Consider a linear constraint

$$
l \leq \sum_{j=1}^{n} a_j x_j \leq u \tag{8.18}
$$

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

$$
\begin{array}{lcccl}
l & \leq & \sum_{j=1}^{n} a_j^{\mathrm{n}} x_j - \sum_{j=1}^{n} \sigma_j y_j, & & \\
 & & \sum_{j=1}^{n} a_j^{\mathrm{n}} x_j + \sum_{j=1}^{n} \sigma_j y_j & \leq & u, \\
0 & \leq & x_j + y_j, & & \\
0 & \leq & -x_j + y_j, & & j = 1, ..., n.
\end{array}
\tag{8.19}
$$

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 uncertain constraint (8.18) with its ellipsoidal robust counterpart

$$\begin{array}{rcl} l & \leq & \sum\limits_{j=1}^{n} a_j^{\rm n} x_j - \Omega z, \\ & & \sum\limits_{j=1}^{n} a_j^{\rm n} x_j + \Omega z \quad \leq \quad u, \\ 0 & \leq & z - \sqrt{\sum\limits_{j=1}^{n} \sigma_j^2 x_j^2}. \end{array} \tag{8.20}$$

Note that with the model of random perturbations, a vector x satisfying (8.20) satisfies a realization of (8.18) 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 to say that x satisfies *all* realizations of (8.18). On the other hand, the uncertainty set associated with (8.19) is the box

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

while the uncertainty set associated with (8.20) is the ellipsoid

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

For a moderate value of Ω, say, $\Omega = 6$, and $n \geq 40$, the ellipsoid $E(\Omega)$ in its diameter, typical linear sizes, volume, etc., is incomparably less than the box B, the difference becoming the more dramatic the larger is the dimension n of the box and the ellipsoid. It follows that the ellipsoidal robust counterpart (8.20) of the randomly perturbed uncertain constraint (8.18) is much less conservative than the interval robust counterpart (8.19), while ensuring basically the same "robustness guarantees". To illustrate this important point, consider a numerical example as follows:

> 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 for different assets are independent of each other. The problem is to distribute \$ 1 among the assets in order to get the largest possible total return on the resulting portfolio.

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

$$\begin{array}{lrcl} \text{maximize} & \sum\limits_{j=1}^{n} a_j x_j & & \\ \text{subject to} & \sum\limits_{j=1}^{n} x_j & = & 1, \\ 0 \leq & x_j, & & j = 1, ..., n. \end{array} \tag{8.21}$$

where a_j are the uncertain returns of the assets. Both the nominal optimal solution (set all returns a_j equal to their nominal values δ_j) and the risk-neutral Stochastic Programming approach (maximize the expected total return) result in the same solution: our \$ 1 should be invested in the most promising asset(s) – the one(s) with the maximal nominal return. This solution, however, can be very "unreliable" if, as it typically is the case in reality, the most promising asset has the largest volatility σ and is in this sense the most risky. To reduce risk, one can use the Robust Counterpart approach which results in the optimization problems

$$\begin{array}{llcll} \text{maximize} & & & t & \\ \text{subject to} & 0 & \leq & -t+\sum_{j=1}^{n}(\delta_j-\sigma_j)x_j, & \\ & & & \sum_{j=1}^{n}x_j & = \quad 1, \\ & 0 & \leq & x_j, & j=1,...,n. \end{array} \tag{8.22}$$

(the interval model of uncertainty) and

$$\begin{array}{llcll} \text{maximize} & & & t & \\ \text{subject to} & 0 & \leq & -t+\sum_{j=1}^{n}(\delta_j)x_j-\Omega z, & \\ & 0 & \leq & z-\sqrt{\sum_{j=1}^{n}\sigma_j^2x_j^2}, & \\ & & & \sum_{j=1}^{n}x_j & = \quad 1, \\ & 0 & \leq & x_j, & j=1,...,n. \end{array} \tag{8.23}$$

(the ellipsoidal model of uncertainty; note that (8.23) is, essentially, the risk-averted portfolio model proposed in mid-50's by Markowitz).

The solution of (8.22) is evident – our \$ 1 should be invested into 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 (8.23) prescribes a quite reasonable diversification of investments which allows to get much better total return than (8.22), with, basically, zero risk[3]. To get an illustration, 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:

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

---

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

Here is a MATLAB script which builds the associated problem (8.23), 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 for the resulting portfolio (in these simulations, the returns of all assets are uniformly distributed in the corresponding intervals):

```
% File: rlo2.m

% Problem:
%
% maximize t subject to
% t <= sum(delta(j)*x(j)) -Omega*z,
% y(j) = sigma(j)*x(j), j=1,...,n,
% sum(x(j)) = 1,
% norm(y) <= z,
% 0 <= x.

clear prob;
n     = 300;
Omega = 6;

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

% setting mosekopt description of the problem

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

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

% displaying the solution
xx = res.sol.itr.xx;
t  = xx(1);

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

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

% running simulations

Nsim = 10000;
out  = zeros(Nsim,1);
for i=1:Nsim,
    returns  = delta+(2*rand(1,n)-1).*sigma;
    out(i)   = returns*x;
end;
disp(sprintf('Actual returns over %d simulations:',Nsim));
disp(sprintf('Min=%5.4f Mean=%5.4f Max=%5.4f StD=%5.2f',...
    min(out),mean(out),max(out),std(out)));
hist(out);
```

Here are the results displayed by the script:

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

Note that with our setup, there is exactly one asset with guaranteed return greater than 1 – asset # 1 (bank savings, return 1.04, zero volatility). Consequently, the interval robust counterpart (8.22) prescribes to put our $ 1 to bank, thus getting 4%-profit. In contrast to this, the diversified portfolio given by the optimal solution of (8.23) *never* yields profit less than 57.2 %, and yields at average 69.67%-profit with pretty low (0.03) standard deviation. We

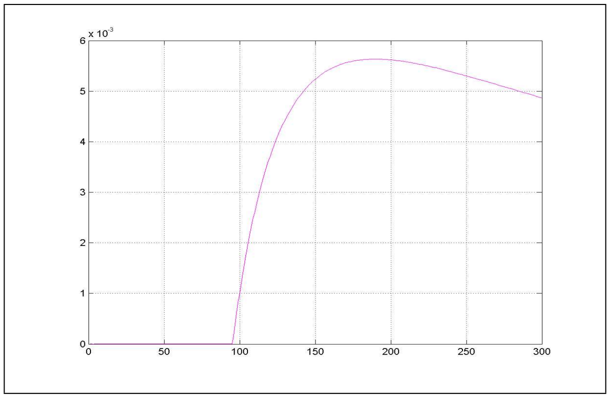

Figure 8.1: Distribution of investments among the assets in the optimal solution of.

see that *in favorable circumstances the ellipsoidal robust counterpart of an uncertain linear program indeed is less conservative than, although is 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 nominal return 2.00 and volatility 1.152). Assuming that the actual return is uniformly distributed in the corresponding interval and running 10,000 simulations, we get the results as follows:

```
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.

8.1.4.1.1 Combined Interval-Ellipsoidal Robust Counterpart. We have considered the case when the coefficients a_j of uncertain linear constraint (8.18) are affected by independent of each other random perturbations symmetrically distributed in given intervals $[-\sigma_j, \sigma_j]$ and have discussed two ways to model the uncertainty:

- *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 RC (8.19);
- *ellipsoidal uncertainty model* ($\mathcal{U}$ is the ellipsoid $E(\Omega)$), which does take into account the stochastic nature of data perturbations and yields the ellipsoidal RC (8.20).

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. For example, the ellipsoid $E(6)$, although much more "narrow" than B in most of the directions, is 6 times "thicker" 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 are forbidden by our model of perturbations, so that 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 $\mathcal{U}(\Omega) = E(\Omega) \bigcap B$?

First, it turns out that the RC of (8.18) corresponding to the uncertainty set $\mathcal{U}(\Omega)$ is still given by a system of linear and conic quadratic inequalities, specifically, the system

$$
\begin{array}{rclcl}
l & \leq & \sum\limits_{j=1}^{n} a_j^{\mathrm{n}} x_j - \sum\limits_{j=1}^{n} \sigma_j y_j - \Omega \sqrt{\sum\limits_{j=1}^{n} \sigma_j^2 u_j^2}, & & \\
 & & \sum\limits_{j=1}^{n} a_j^{\mathrm{n}} x_j + \sum\limits_{j=1}^{n} \sigma_j z_j + \Omega \sqrt{\sum\limits_{j=1}^{n} \sigma_j^2 v_j^2} & \leq & u, \\
-y_j & \leq & x_j - u_j & \leq & y_j,\ j = 1, ..., n, \\
-z_j & \leq & x_j - v_j & \leq & z_j,\ j = 1, ..., n.
\end{array}
\tag{8.24}
$$

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 (8.24), then the probability for x to satisfy a realization of (8.18) 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, as the robust version of the constraint, system of linear and conic quadratic inequalities (8.24). 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, and, consequently, use in the RC different "robust versions" of different original uncertain constraints.

8.1.5 Further references

For further information about robust linear optimization consult [14, 15].

8.2 Geometric (posynomial) optimization

8.2.1 The problem

A *geometric optimization* problem can be stated as follows

$$\begin{array}{lll} \text{minimize} & \sum_{k \in J_0} c_k \prod_{j=0}^{n-1} t_j^{a_{kj}} & \\ \text{subject to} & \sum_{k \in J_i} c_k \prod_{j=0}^{n-1} t_j^{a_{kj}} \leq 1, & i = 1, \ldots, m, \\ & t > 0, & \end{array} \tag{8.25}$$

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 (8.25) 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

$$t_j = e^{x_j} \tag{8.26}$$

we obtain the problem

$$\begin{array}{lll} \text{minimize} & \sum_{k \in J_0} c_k e^{\sum_{j=0}^{n-1} a_{kj} x_j} & \\ \text{subject to} & \sum_{k \in J_i} c_k e^{\sum_{j=0}^{n-1} a_{kj} x_j} \leq 1, & i = 1, \ldots, m, \end{array} \tag{8.27}$$

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

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

for a term and hence the number of terms is T.

As stated, the problem (8.27) is non-separable. However, using

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

we obtain the separable problem

$$\begin{array}{lrcll} \text{minimize} & \sum\limits_{t\in J_0} e^{v_t} & & & \\ \text{subject to} & \sum\limits_{t\in J_i} e^{v_t} & \leq & 1, & i=1,\ldots,m, \\ & \sum\limits_{j=0}^{n-1} a_{tj}x_j - v_t & = & -\log(c_t), & t=0,\ldots,T, \end{array} \tag{8.28}$$

which is a separable convex optimization problem.

One warning about this approach is that the function

$$e^x$$

is only well-defined for small values of x in absolute value. Indeed e^x grows very rapidly as x becomes larger. Therefore numerical problems may arise when solving the problem on this form.

8.2.2 Applications

A large number of practical applications, particularly in electrical circuit design, can be cast as a geometric optimization problem. We will not review those applications here but rather we refer the reader to [16] and the references therein.

8.2.3 Modelling tricks

A lot of tricks that can be used modelling posynomial optimization problems are described in [16]. Therefore, in this section we cover only one important case.

8.2.3.1 Equalities

In general equalities are not allowed in (8.25), i.e.

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

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

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

of a monomial equality. The equality is identical to

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

which in turn is identical to the two inequalities

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

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

8.2.4 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.

8.2.4.1 Finite unattainable solution

The following problem illustrates an unattainable solution:

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

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

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

We now see that the optimal solution implies that $t_x = -\infty$ or $t_y = -\infty$, which is unattainable.

It should now be clear what the issue is: 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.

8.2.4.2 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:

$$\begin{array}{lll} \text{minimize} & x^{-2} & \\ \text{subject to} & x^{-1} & \leq \ 1, \\ & x > 0, & \end{array}$$

which is a valid geometric programming problem. In this case the optimal objective is 0, but this requires $x = \infty$, which is unattainable.

Again, this specific case will appear if a variable x appears only with negative exponents in the problem, implying that each term in which it appears can be minimized for $x \rightarrow \infty$.

8.2.5 An example

Consider the example

$$\begin{array}{lccc} \text{minimize} & x^{-1}y & & \\ \text{subject to} & x^2y^{-\frac{1}{2}} + 3y^{\frac{1}{2}}z^{-1} & \leq & 1, \\ & xy^{-1} & = & z^2, \\ & -x & \leq & -\frac{1}{10}, \\ & x & \leq & 3, \\ & x, y, z > 0, & & \end{array}$$

which is not a geometric optimization problem. However, using the obvious transformations we obtain the problem

$$\begin{array}{lccc} \text{minimize} & x^{-1}y & & \\ \text{subject to} & x^2y^{-\frac{1}{2}} + 3y^{\frac{1}{2}}z^{-1} & \leq & 1, \\ & xy^{-1}z^{-2} & \leq & 1, \\ & x^{-1}yz^2 & \leq & 1, \\ & \frac{1}{10}x^{-1} & \leq & 1, \\ & \frac{1}{3}x & \leq & 1, \\ & x, y, z > 0, & & \end{array} \tag{8.29}$$

which is a geometric optimization problem.

8.2.6 Solving the example

The problem 8.29 can be defined and solved in the MOSEK toolbox as shown below.

```
% go2.m

c     = [1 1 3 1 1 0.1 1/3]';
a     = sparse([[-1  1  0];
                [2 -0.5 0];
                [0 0.5 -1];
                [1 -1 -2];
                [-1 1 2];
                [-1 0 0];
                [1 0 0]]);

map   = [0 1 1 2 3 4 5]';
[res] = mskgpopt(c,a,map);

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

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

% Add appropriate terms together.
f = sparse(map+1,1:7,ones(size(map)))*v;

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

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

8.2.7 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 command line tool `mskexpopt`. See the MOSEK Tools User's manual for details on `mskexpopt`. This file format can be useful for sending debug information to MOSEK or testing. It's also possible to read the above format with the command:

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

8.2.8 Further information

More information about geometric optimization problems is located in [12, 13, 16].

Chapter 9

Modelling

In this chapter we will discuss the following issues:

- The formal definitions of the problem types that MOSEK can solve.
- The solution information produced by MOSEK.
- The information produced by MOSEK if the problem is infeasible.
- A set of examples showing different ways of formulating commonly occurring problems so that they can be solved by MOSEK.
- Recommendations for formulating optimization problems.

9.1 Linear optimization

A linear optimization problem can be written as

$$\begin{array}{lccccl} \mbox{minimize} & & & c^T x + c^f & & \\ \mbox{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \end{array} \tag{9.1}$$

where

- m is the number of constraints.
- n is the number of decision variables.
- $x \in R^n$ is a vector of decision variables.
- $c \in R^n$ is the linear part of the objective function.
- $A \in R^{m \times n}$ is the constraint matrix.

- $l^c \in R^m$ is the lower limit[1] on the activity for the constraints.
- $u^c \in R^m$ is the upper limit on the activity for the constraints.
- $l^x \in R^n$ is the lower limit on the activity for the variables.
- $u^x \in R^n$ is the upper limit on the activity for the variables.

A primal solution (x) is *(primal) feasible* if it satisfies all constraints in (9.1). If (9.1) has at least one primal feasible solution, then (9.1) is said to be (primal) feasible.

In case (9.1) does not have a feasible solution, the problem is said to be *(primal) infeasible*.

9.1.1 Duality for linear optimization

Corresponding to the primal problem (9.1), there is a dual problem

$$\begin{array}{lccl} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c & & \\ & +(l^x)^T s_l^x - (u^x)^T s_u^x + c^f & & \\ \text{subject to} & A^T y + s_l^x - s_u^x & = & c, \\ & -y + s_l^c - s_u^c & = & 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0. & & \end{array} \tag{9.2}$$

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. For example

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

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 (9.2). If (9.2) has at least one feasible solution, then (9.2) is *(dual) feasible*, otherwise the problem is *(dual) infeasible*.

We will denote a solution

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

so that x is a solution to the primal problem (9.1), and

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

is a solution to the corresponding dual problem (9.2). A solution which is both primal and dual feasible is denoted a *primal-dual feasible primal-dual* solution.

---

[1] We will use the words "bound" and "limit" interchangeably.

9.1.1.1 A primal-dual feasible solution

Let

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

be a primal-dual feasible solution, and let

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

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

$$\begin{array}{c} c^T x^* + c^f - ((l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f) \\ = \sum\limits_{i=1}^{m} ((s_l^c)_i^* ((x_i^c)^* - l_i^c) + (s_u^c)_i^* (u_i^c - (x_i^c)^*) + \sum\limits_{j=1}^{n} ((s_l^x)_j^* (x_j - l_j^x) + (s_u^x)_j^* (u_j^x - x_j^*)) \\ \geq 0 \end{array}$$

where the first relation can be obtained by multiplying the dual constraints (9.2) by x and x^c respectively, and the second relation comes from the fact that each term in each sum is nonnegative. It follows that the primal objective will always be greater than or equal to the dual objective.

We then define the *duality gap* as the difference between the primal objective value and the dual objective value, i.e.

$$c^T x^* + c^f - ((l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f)$$

Please note that the duality gap will always be nonnegative.

9.1.1.2 An optimal solution

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

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

are satisfied.

If (9.1) 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.

9.1.1.3 Primal infeasible problems

If the problem (9.1) is infeasible (has no feasible solution), MOSEK will report a primal certificate of the infeasibility: The dual solution reported is a certificate of infeasibility, and the primal solution is undefined.

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

$$\begin{array}{lccc} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x & & \\ \text{subject to} & A^T y + s_l^x - s_u^x & = & 0, \\ & -y + s_l^c - s_u^c & = & 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0. & & \end{array} \tag{9.3}$$

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

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

to (9.3) so that

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

Such a solution implies that (9.3) is unbounded, and that its dual is infeasible.

We note that the dual of (9.3) is a problem whose constraints are identical to the constraints of the original primal problem (9.1): If the dual of (9.3) is infeasible, so is the original primal problem.

9.1.1.4 Dual infeasible problems

If the problem (9.2) is infeasible (has no feasible solution), MOSEK will report a dual certificate of the infeasibility: The primal solution reported is a certificate of infeasibility, and the dual solution is undefined.

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

$$\begin{array}{lccccc} \text{minimize} & & & c^T x & & \\ \text{subject to} & & & Ax - x^c & = & 0, \\ & \bar{l}^c & \leq & x^c & \leq & \bar{u}^c, \\ & \bar{l}^x & \leq & x & \leq & \bar{u}^x \end{array} \tag{9.4}$$

where

$$\bar{l}_i^c = \left\{ \begin{array}{ll} 0, & \text{if } l_i^c > -\infty, \\ -\infty & \text{otherwise} \end{array} \right. \quad \text{and} \quad \bar{u}_i^c := \left\{ \begin{array}{ll} 0, & \text{if } u_i^c < \infty, \\ \infty & \text{otherwise} \end{array} \right.$$

and

$$\bar{l}_j^x = \left\{ \begin{array}{ll} 0, & \text{if } l_j^x > -\infty, \\ -\infty & \text{otherwise} \end{array} \right. \quad \text{and} \quad \bar{u}_j^x := \left\{ \begin{array}{ll} 0, & \text{if } u_j^x < \infty, \\ \infty & \text{otherwise} \end{array} \right.$$

so that the objective value $c^T x$ is negative. Such a solution implies that (9.4) is unbounded, and that dual of (9.4) is infeasible.

We note that the dual of (9.4) is a problem whose constraints are identical to the constraints of the original dual problem (9.2): If the dual of (9.4) is infeasible, so is the original dual problem.

9.1.2 Primal and dual infeasible case

In case that both the primal problem (9.1) and the dual problem (9.2) are infeasible, MOSEK will report only one of the two possible certificates — which one is not defined (MOSEK returns the first certificate found).

9.2 Linear network flow problems

Network flow problems are a special class of linear optimization problems which has many applications. The class of network flow problems can be specified as follows. Let $G = (\mathcal{N}, \mathcal{A})$ be a directed network. Associated with every arc $(i, j) \in \mathcal{A}$ is a cost c_{ij} and a capacity $[l^x_{ij}, u^x_{ij}]$. Moreover, associated with each node $i \in \mathcal{N}$ in the network is a lower limit l^c_i and an upper limit u^c_i on the demand(supply) of the node. Now the minimum cost of a network flow problem can be stated as follows

$$\begin{array}{lccccl}\text{minimize} & & & \displaystyle\sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} & & & \\ \text{subject to} & l^c_i & \leq & \displaystyle\sum_{\{j:(i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j:(j,i)\in\mathcal{A}\}} x_{ji} & \leq & u^c_i & \forall i \in \mathcal{N}, \\ & l^x_{ij} & \leq & x_{ij} & \leq & u^x_{ij} & \forall (i,j) \in \mathcal{A}.\end{array} \tag{9.5}$$

A classical example of a network flow problem is the transportation problem, where the objective is to distribute goods from warehouses to customers at lowest possible total cost, see [2] for a detailed application reference.

It is well-known that problems with network flow structure can be solved efficiently with a specialized version of the simplex method. MOSEK includes a highly tuned network simplex implementation, see Section 10.3.1 for further details on how to invoke the network optimizer.

9.3 Quadratic and quadratically constrained optimization

A convex quadratic optimization problem is an optimization problem of the form

$$\begin{array}{lccccll}\text{minimize} & & & \frac{1}{2}x^TQ^ox + c^Tx + c^f & & & \\ \text{subject to} & l^c_k & \leq & \frac{1}{2}x^TQ^kx + \displaystyle\sum_{j=0}^{n-1} a_{k,j}x_j & \leq & u^c_k, & k = 0,\ldots,m-1, \\ & l^x & \leq & x & \leq & u^x, & j = 0,\ldots,n-1,\end{array} \tag{9.6}$$

where the convexity requirement implies that

- Q^o is a symmetric positive semi-definite matrix.
- If $l_k^c = -\infty$, then Q^k is a symmetric positive semi-definite matrix.
- If $u_k^c = \infty$, then Q^k is a symmetric negative semi-definite matrix.
- If $l_k > -\infty$ and $u_k^k < \infty$, then Q^k is a zero matrix.

The convexity requirement is very important and it is strongly recommended that MOSEK is applied to convex problems only.

9.3.1 A general recommendation

Any convex quadratic optimization problem can be reformulated as a conic optimization problem. It is our experience that for the majority of practical applications it is better to cast them as conic problems because

- the resulting problem is convex by construction, and
- the conic optimizer is more efficient than the optimizer for general quadratic problems.

See Section 9.4.4.1 for further details.

9.3.2 Reformulating as a separable quadratic problem

The simplest quadratic optimization problem is

$$\begin{array}{lccl} \text{minimize} & 1/2x^TQx + c^Tx & & \\ \text{subject to} & Ax & = & b, \\ & x \geq 0. & & \end{array} \tag{9.7}$$

The problem (9.7) is said to be a separable problem if Q is a diagonal matrix or, in other words, that the quadratic terms in the objective all have this form

$$x_j^2$$

instead of this form

$$x_j x_i.$$

The separable form has the following advantages:

- It is very easy to check the convexity assumption, and
- the simpler structure in a separable problem usually makes it easier to solve.

It is well-known that a positive semi-definite matrix Q can always be factorized, i.e. a matrix F exists so that

$$Q = F^T F. \tag{9.8}$$

In many practical applications of quadratic optimization F is known explicitly; for example if Q is a covariance matrix, F would be the set of observations producing it.

Using (9.8), the problem (9.7) can be reformulated as

$$\begin{array}{lrcl} \text{minimize} & 1/2y^T Iy + c^T x & & \\ \text{subject to} & Ax & = & b, \\ & Fx - y & = & 0, \\ & x \geq 0. & & \end{array} \tag{9.9}$$

The problem (9.9) is also a quadratic optimization problem and has more constraints and variables than (9.7). However, the problem is separable. Normally, if F has fewer rows than columns, it is worthwhile to reformulate as a separable problem. Indeed consider the extreme case where F has one dense row and hence Q will be dense matrix.

The idea presented above is applicable to quadratic constraints too. Now consider the constraint

$$1/2x^T(F^T F)x \leq b \tag{9.10}$$

where F is a matrix and b is a scalar. (9.10) can be reformulated as

$$\begin{array}{rcl} 1/2y^T Iy & \leq & b, \\ Fx - y & = & 0. \end{array}$$

It should be obvious how to generalize this idea to make any convex quadratic problem separable.

Next, consider the constraint

$$1/2x^T(D + F^T F)x \leq b$$

where D is a positive semi-definite matrix, F is a matrix, and b is a scalar. We assume that D has a simple structure, i.e. D is for instance a diagonal or a block diagonal matrix. If this is the case, then it may be worthwhile performing the reformulation

$$\begin{array}{rcl} 1/2((x^T Dx) + y^T Iy) & \leq & b, \\ Fx - y & = & 0. \end{array}$$

Now the question may arise: When should a quadratic problem be reformulated to make it separable or near separable? The simplest rule of thumb is that it should be reformulated if the number of non-zeros used to represent the problem decreases when reformulating the problem.

9.4 Conic optimization

Conic optimization can be seen as a generalization of linear optimization. Indeed a conic optimization problem is a linear optimization problem plus a constraint of the form

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

where $\mathcal{C}$ is a convex cone. A complete conic problem has the form

$$\begin{array}{lccccc} \text{minimize} & & & c^T x + c^f & & \\ \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \\ & & & x \in \mathcal{C}. & & \end{array} \tag{9.11}$$

The cone $\mathcal{C}$ can be a Cartesian product of p convex cones, i.e.

$$\mathcal{C} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_p$$

in which case $x \in \mathcal{C}$ can be written as

$$x = (x_1, \ldots, x_p),\ x_1 \in \mathcal{C}_1, \ldots, x_p \in \mathcal{C}_p$$

where each $x_t \in R^{n_t}$. Please note that the n-dimensional Euclidean space R^n is a cone itself, so simple linear variables are still allowed.

MOSEK supports only a limited number of cones, specifically

$$\mathcal{C} = \mathcal{C}_1 \times \cdot \times \mathcal{C}_p$$

where each $\mathcal{C}_t$ has one of the following forms

- R set:
$$\mathcal{C}_t = \{x \in R^{n^t}\}.$$
- Quadratic cone:
$$\mathcal{C}_t = \left\{ x \in R^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\}.$$
- Rotated quadratic cone:
$$\mathcal{C}_t = \left\{ x \in R^{n^t} : 2x_1 x_2 \geq \sum_{j=3}^{n^t} x_j^2,\ x_1, x_2 \geq 0 \right\}.$$

Although these cones may seem to provide only limited expressive power they can be used to model a large range of problems as demonstrated in Section 9.4.4.

9.4.1 Duality for conic optimization

The dual problem corresponding to the conic optimization problem (9.11) is given by

$$
\begin{array}{lrcl}
\text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c & & \\
 & +(l^x)^T s_l^x - (u^x)^T s_u^x + c^f & & \\
\text{subject to} & A^T y + s_l^x - s_u^x + s_n^x & = & c, \\
 & -y + s_l^c - s_u^c & = & 0, \\
 & s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0, \\
 & s_n^x \in \mathcal{C}^* & &
\end{array}
\tag{9.12}
$$

where the dual cone $\mathcal{C}^*$ is a product of cones

$$\mathcal{C}^* = \mathcal{C}_1^* \times \cdots \mathcal{C}_p^*$$

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:

- R set:
$$\mathcal{C}_t = \left\{ x \in R^{n^t} \right\} \quad \Leftrightarrow \quad \mathcal{C}_t^* := \left\{ s \in R^{n^t} \ : \ s = 0 \right\}.$$
- Quadratic cone:
$$\mathcal{C}_t := \left\{ x \in R^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\} \quad \Leftrightarrow \quad \mathcal{C}_t^* = \mathcal{C}_t.$$
- Rotated quadratic cone:
$$\mathcal{C}_t := \left\{ x \in R^{n^t} : 2x_1x_2 \geq \sum_{j=3}^{n^t} x_j^2, \ x_1, x_2 \geq 0 \right\}. \quad \Leftrightarrow \quad \mathcal{C}_t^* = \mathcal{C}_t.$$

9.4.2 The dual of the dual

The dual problem corresponding to the dual problem is the primal problem.

9.4.3 Infeasibility

In case MOSEK finds a problem to be infeasible it will report a certificate of the infeasibility. This works exactly as for linear problems (see sections 9.1.1.3 and 9.1.1.4).

9.4.4 Examples

This section contains several examples of inequalities and problems that can be cast as conic optimization problems.

9.4.4.1 Quadratic objective and constraints

From Section 9.3.2 we know that any convex quadratic problem can be stated on the form

$$\begin{array}{lll} \text{minimize} & 0.5\left\|Fx\right\|^2 + c^T x, & \\ \text{subject to} & 0.5\left\|Gx\right\|^2 + a^T x & \leq \quad b, \end{array} \tag{9.13}$$

where F and G are matrices and c and a are vectors. For simplicity we assume that there is only one constraint, but it should be obvious how to generalize the methods to an arbitrary number of constraints.

Problem (9.13) can be reformulated as

$$\begin{array}{llll} \text{minimize} & 0.5\left\|t\right\|^2 + c^T x, & & \\ \text{subject to} & 0.5\left\|z\right\|^2 + a^T x & \leq & b, \\ & Fx - t & = & 0, \\ & Gx - z & = & 0 \end{array} \tag{9.14}$$

after the introduction of the new variables t and z. It is easy to convert this problem to a conic quadratic optimization problem, i.e.

$$\begin{array}{lllll} \text{minimize} & v + c^T x, & & & \\ \text{subject to} & p + a^T x & = & b, & \\ & Fx - t & = & 0, & \\ & Gx - z & = & 0, & \\ & w & = & 1, & \\ & q & = & 1, & \\ & \left\|t\right\|^2 & \leq & 2vw, & v, w \geq 0, \\ & \left\|z\right\|^2 & \leq & 2pq, & p, q \geq 0. \end{array} \tag{9.15}$$

In this case we can model the last two inequalities using rotated quadratic cones.

If we assume that F is a non-singular matrix — for instance a diagonal matrix — then

$$x = F^{-1} t.$$

and hence we can eliminate x from the problem to obtain:

$$\begin{array}{lllll} \text{minimize} & v + c^T F^{-1} t, & & & \\ \text{subject to} & p + a^T F^{-1} t & = & b, & \\ & V F^{-1} t - z & = & 0, & \\ & w & = & 1, & \\ & q & = & 1, & \\ & \left\|t\right\|^2 & \leq & 2vw, & v, w \geq 0, \\ & \left\|z\right\|^2 & \leq & 2pq, & p, q \geq 0. \end{array} \tag{9.16}$$

In most cases MOSEK will perform this reduction automatically during the presolve phase before the optimization is performed.

9.4.4.2 Minimizing a sum of norms

The next example is the problem of minimizing a sum of norms i.e. the problem

$$\begin{array}{lrcl} \text{minimize} & \sum_{i=1}^{k} \left\|x^i\right\| & & \\ \text{subject to} & Ax & = & b, \end{array} \tag{9.17}$$

where

$$x := \begin{bmatrix} x^1 \\ \vdots \\ x^k \end{bmatrix}.$$

This problem is equivalent to

$$\begin{array}{lrcll} \text{minimize} & \sum_{i=1}^{k} z_i & & & \\ \text{subject to} & Ax & = & b, & \\ & \left\|x^i\right\| & \leq & z_i, & i = 1, \ldots, k, \end{array} \tag{9.18}$$

which in turn is equivalent to

$$\begin{array}{lrcl} \text{minimize} & \sum_{i=1}^{k} z_i & & \\ \text{subject to} & Ax & = & b, \\ & (z_i, x^i) \in \mathcal{C}_i, & & i = 1, \ldots, k \end{array} \tag{9.19}$$

where all $\mathcal{C}^i$ are of the quadratic type, i.e.

$$\mathcal{C}_i := \left\{(z_i, x^i) : \ z_i \geq \left\|x^i\right\|\right\}.$$

The dual problem corresponding to (9.19) is

$$\begin{array}{lrcll} \text{maximize} & b^T y & & & \\ \text{subject to} & A^T y + s & = & c, & \\ & t_i & = & 1, & i = 1, \ldots, k, \\ & (t_i, s^i) \in \mathcal{C}_i, & & & i = 1, \ldots, k \end{array} \tag{9.20}$$

where

$$s := \begin{bmatrix} s^1 \\ \vdots \\ s^k \end{bmatrix}.$$

This problem is equivalent to

$$\begin{array}{lrcll} \text{maximize} & b^T y & & & \\ \text{subject to} & A^T y + s & = & c, & \\ & \left\|s^i\right\|_2^2 & \leq & 1, & i = 1, \ldots, k. \end{array} \tag{9.21}$$

Please note that the dual problem can be reduced to an "ordinary" convex quadratically constrained optimization problem in this case due to the special structure of the primal problem. In some cases it turns out that it is much better to solve the dual problem (9.20) rather than the primal problem (9.19).

9.4.4.3 Modelling polynomial terms using conic optimization

Generally an arbitrary polynomial term of the form

$$fx^g$$

cannot be represented with conic quadratic constraints, however in the following we will demonstrate some special cases where it is possible.

A particular simple polynomial term is the reciprocal, i.e.

$$\frac{1}{x}.$$

Now, a constraint of the form

$$\frac{1}{x} \leq y$$

where it is required that $x > 0$ is equivalent to

$$1 \leq xy \text{ and } x > 0$$

which in turn is equivalent to

$$\begin{array}{rcl} z & = & \sqrt{2}, \\ z^2 & \leq & 2xy. \end{array}$$

The last formulation is a conic constraint plus a simple linear equality.

For example, consider the problem

$$\begin{array}{lccc} \text{minimize} & c^T x & & \\ \text{subject to} & \sum\limits_{j=1}^{n} \frac{f_j}{x_j} & \leq & b, \\ & x \geq 0, & & \end{array}$$

where it is assumed that $f_j > 0$ and $b > 0$. This problem is equivalent to

$$\begin{array}{lcccl} \text{minimize} & c^T x & & & \\ \text{subject to} & \sum\limits_{j=1}^{n} z_j & = & b, & \\ & v_j & = & \sqrt{2}, & j = 1, \ldots, n, \\ & v_j^2 & \leq & 2 z_j x_j, & j = 1, \ldots, n, \\ & x, z \geq 0, & & & \end{array} \tag{9.22}$$

because

$$v_j^2 = 2 \leq 2 z_j x_j$$

implies that

$$\frac{1}{x_j} \leq z_j \text{ and } \sum_{j=1}^{n} \frac{f_j}{x_j} \leq \sum_{j=1}^{n} f_j z_j = b.$$

The problem (9.22) is a conic quadratic optimization problem having n 3 dimensional rotated quadratic cones.

The next example is the constraint

$$\begin{array}{rcl} \sqrt{x} & \geq & |t|, \\ x & \geq & 0, \end{array}$$

where both t and x are variables. This set is identical to the set

$$\begin{array}{rcl} t^2 & \leq & 2xz, \\ z & = & 0.5, \\ x, z, & \geq & 0. \end{array} \tag{9.23}$$

Occasionally when modelling the *market impact* term in portfolio optimization, the polynomial term $x^{\frac{3}{2}}$ occurs. Therefore, consider the set defined by the inequalities

$$\begin{array}{rcl} x^{1.5} & \leq & t, \\ 0 & \leq & x. \end{array} \tag{9.24}$$

We will exploit that $x^{1.5} = x^2/\sqrt{x}$. First define the set

$$\begin{array}{rcl} x^2 & \leq & 2st, \\ s, t & \geq & 0. \end{array} \tag{9.25}$$

Now, if we can make sure that

$$2s \leq \sqrt{x},$$

then we have the desired result since this implies that

$$x^{1.5} = \frac{x^2}{\sqrt{x}} \leq \frac{x^2}{2s} \leq t.$$

Please note that s can be chosen freely and that $\sqrt{x} = 2s$ is a valid choice.

Let

$$\begin{array}{rcl} x^2 & \leq & 2st, \\ w^2 & \leq & 2vr, \\ x & = & v, \\ s & = & w, \\ r & = & \frac{1}{8}, \\ s, t, v, r & \geq & 0, \end{array} \tag{9.26}$$

then

$$\begin{array}{rcl} s^2 & = & w^2 \\ & \leq & 2vr \\ & \leq & \frac{v}{4} \\ & = & \frac{x}{4}. \end{array}$$

Moreover,

$$\begin{array}{rcl} x^2 & \leq & 2st, \\ & \leq & 2\sqrt{\frac{x}{4}}t \end{array}$$

leading to the conclusion

$$x^{1.5} \leq t.$$

(9.26) is a conic reformulation which is equivalent to (9.24). Please note that the $x \geq 0$ constraint does not appear explicitly in (9.25) and (9.26), but implicitly since $x = v \geq 0$.

Finally, it should be mentioned that any polynomial term of form x^g where g is a positive rational number can be represented using conic quadratic constraints [3, pp. 12-13]

9.4.4.4 Further reading

If you want to know more about what can be modelled as a conic optimization problem we recommend the references [20, 15, 3].

9.4.5 Potential pitfalls in conic optimization

While a linear optimization problem either has a bounded optimal solution or is infeasible, the conic case is not as simple as that.

9.4.5.1 Non-attainment in the primal problem

Consider the example

$$\begin{array}{lrcl} \text{minimize} & z & & \\ \text{subject to} & 2yz & \geq & x^2, \\ & x & = & \sqrt{2}, \\ & y, z & \geq & 0. \end{array} \tag{9.27}$$

which corresponds to the problem

$$\begin{array}{lrcl} \text{minimize} & \frac{1}{y} & & \\ \text{subject to} & y & \geq & 0. \end{array} \tag{9.28}$$

Clearly, the optimal objective value is zero but it is never attained because implicitly we assume that the optimal y should be finite.

9.4.5.2 Non-attainment in the dual problem

Next, consider the example

$$\begin{array}{lccl} \text{minimize} & x_4 & & \\ \text{subject to} & x_3 + x_4 & = & 1, \\ & x_1 & = & 0, \\ & x_2 & = & 1, \\ & 2x_1x_2 & \geq & x_3^2, \\ & x_1x_2 & \geq & 0. \end{array} \tag{9.29}$$

which has the optimal solution

$$x_1^* = 0,\ x_2^* = 1,\ x_3^* = 0 \text{ and } x_4^* = 1$$

implying that the optimal primal objective value is 1.

Now, the dual problem corresponding to (9.29) is

$$\begin{array}{lccl} \text{maximize} & y_1 + y_3 & & \\ \text{subject to} & y_2 + s_1 & = & 0, \\ & y_3 + s_2 & = & 0, \\ & y_1 + s_3 & = & 0, \\ & y_1 & = & 1, \\ & 2s_1s_2 & \geq & s_3^2, \\ & s_1s_2 & \geq & 0. \end{array} \tag{9.30}$$

Therefore,

$$y_1^* = 1$$

and

$$s_3^* = -1.$$

This implies that

$$2s_1^*s_2^* \geq (s_3^*)^2 = 1$$

and hence $s_2^* > 0$. Given this fact we can conclude that

$$\begin{array}{rcl} y_1^* + y_3^* & = & 1 - s_2^* \\ & < & 1 \end{array}$$

implying that the optimal dual objective value is 1, however this is never attained. Hence, there no primal and dual bounded optimal solution that has zero duality gap exists. Of course it is possible to find a primal and dual feasible solution such that the duality gap is close to zero, however, s_1^* will be very large (unless a large duality gap is allowed). This is likely to make the problem (9.29) hard to solve.

An inspection of problem (9.29) reveals the constraint $x_1 = 0$, which implies that $x_3 = 0$. If we either add the redundant constraint

$$x_3 = 0$$

to the problem (9.29) or eliminate x_1 and x_3 from the problem it becomes easy to solve.

9.5 Nonlinear convex optimization

MOSEK is capable of solving smooth convex nonlinear optimization problems of the form

$$\begin{array}{lcccl} \text{minimize} & & f(x) + c^T x & & \\ \text{subject to} & & g(x) + Ax - x^c & = & 0, \\ l^c & \leq & x^c & \leq & u^c, \\ l^x & \leq & x & \leq & u^x, \end{array} \tag{9.31}$$

where

- m is the number of constraints.
- n is the number of decision variables.
- $x \in R^n$ is a vector of decision variables.
- $x^c \in R^m$ is a vector of constraints or slack variables.
- $c \in R^n$ is the linear part objective function.
- $A \in R^{m \times n}$ is the constraint matrix.
- $l^c \in R^m$ is the lower limit[2] on the activity for the constraints.
- $u^c \in R^m$ is the upper limit on the activity for the constraints.
- $l^x \in R^n$ is the lower limit on the activity for the variables.
- $u^x \in R^n$ is the upper limit on the activity for the variables.
- $f : R^n \rightarrow R$ is a nonlinear function.
- $g : R^n \rightarrow R^m$ is a nonlinear vector function.

---

[2] We will use the words “bound” and “limit” interchangeably.

This means that the ith constraint has the form

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

when the x_i^c variable has been eliminated.

The linear term Ax is not included in $g(x)$ since it can be handled much more efficiently as a separate entity when optimizing.

The nonlinear functions f and g must be smooth (twice differentiable) in all $x \in [l^x; u^x]$. Moreover, $f(x)$ must be a convex function and $g_i(x)$ must satisfy

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

9.5.1 Duality

So far, we have not discussed what happens when MOSEK is used to solve a primal or dual infeasible problem. In the subsequent section these issues are addressed.

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

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

and the dual problem is given by

$$\begin{array}{lccc} \text{maximize} & L(x^c, x, y, s_l^c, s_u^c, s_l^x, s_u^x) & & \\ \text{subject to} & \nabla_{(x^c,x)} L(x^c, x, y, s_l^c, s_u^c, s_l^x, s_u^x) & = & 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0. & & \end{array}$$

which is equivalent to

$$\begin{array}{lccc} \text{maximize} & f(x) - y^T g(x) - x^T (\nabla f(x)^T - \nabla g(x)^T y) & & \\ & +((l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f & & \\ \text{subject to} & -\nabla f(x)^T + A^T y + \nabla g(x)^T y + s_l^x - s_u^x & = & c, \\ & -y + s_l^c - s_u^c & = & 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0. & & \end{array} \tag{9.32}$$

9.6 Recommendations

Often an optimization problem can be formulated in several different ways, and the exact formulation used may have a significant impact on the solution time and the quality of the solution. In some cases the difference between a "good" and a "bad" formulation means the ability to solve the problem or not.

Below is a list of several issues that you should be aware of when developing a good formulation.

1. Sparsity is very important. The constraint matrix A is assumed to be a sparse matrix, where sparse means that it contains many zeros (typically less than 10% non-zeros). Normally, when A is sparser, less memory is required to store the problem and it can be solved faster.

2. Avoid large bounds as these can introduce all sorts of numerical problems. Assume that a variable x_j has the bounds
$$0.0 \leq x_j \leq 1.0e16.$$
The number 1.0e16 is large and it is very likely that the constraint $x_j \leq 1.0e16$ is non-binding at optimum, and therefore that the bound $1.0e16$ will not cause problems. Unfortunately, this is a naïve assumption because the bound $1.0e16$ may actually affect the presolve, the scaling, the computation of the dual objective value, etc. In this case the constraint $x_j \geq 0$ is likely to be sufficient, i.e. $1.0e16$ is just a way of representing infinity.

3. Avoid large penalty terms in the objective, i.e. do not have large terms in the linear part of the objective function. They will most likely cause numerical problems.

4. On a computer all computations are performed in finite precision, which implies that
$$1 = 1 + \varepsilon$$
where ε is about 10^{-16}. This means that the results of all computations are truncated leading to the introduction of rounding errors. The upshot is that very small numbers and very large numbers should be avoided, e.g. it is recommended that all elements in A are either zero or belong to the interval $[10^{-6}, 10^{6}]$. The same holds for the bounds and the linear objective.

5. Decreasing the number of variables or constraints does not *necessarily* make it easier to solve a problem. In certain cases, i.e. in nonlinear optimization, it might be a good idea to introduce more constraints and variables if it makes the model separable. Also a big but sparse problem might be advantageous compared to a smaller but denser problem.

6. Try to avoid linearly dependent rows among the linear constraints. Network flow problems and multi-commodity network flow problems, for example, often contain one or more linearly dependent rows.

7. Finally, it is recommended to consult some of the papers about preprocessing to get some ideas about efficient formulations. See e.g. [4, 6, 18, 19].

9.6.1 Avoid nearly infeasible models

Consider the linear optimization problem

$$\begin{array}{lrcl} \text{minimize} & & & \\ \text{subject to} & x+y & \leq & 10^{-10}+\alpha, \\ & 1.0e4x+2.0e4y & \geq & 10^{-6}, \\ & x,y \geq 0. & & \end{array} \tag{9.33}$$

Clearly, the problem is feasible for $\alpha = 0$. However, for $\alpha = -1.0e-10$ the problem is infeasible. This implies that an insignificant change in the right side of the constraints makes the problem status switch from feasible to infeasible. Such a model should be avoided.

9.7 Examples continued

9.7.1 The absolute value

Assume we have a constraint for the form

$$|f^T x + g| \leq b \tag{9.34}$$

where $x \in R^n$ is a vector of variables, and $f \in R^n$ and $g, b \in R$ are constants.

It is easy to verify that the constraint (9.34) is equivalent to

$$-b \leq f^T x + g - t \leq b \tag{9.35}$$

which is a set of ordinary linear inequality constraints.

Please note that equalities involving and absolute value such as

$$|x| = 1$$

cannot be formulated as a linear or even a convex optimization problem. It requires integer optimization.

9.7.2 The Markowitz portfolio model

In this section we will show how to model several versions of the Markowitz portfolio model using conic optimization.

The Markowitz portfolio model deals with the problem of selecting a portfolio of assets i.e. stocks, bonds, etc. The goal is to find a portfolio such that for a given return the risk is minimized. The assumptions are:

- A portfolio can consist of n traded assets numbered $1, 2, \ldots$ held over a period of time.
- w_j^0 is the initial holding of asset j where $\sum_j w_j^0 > 0$.
- r_j is the return on asset j and is assumed to be a random variable. r has known mean $\bar{r}$ and covariance Σ.

The variable x_j denotes the amount of asset j traded in the given period of time and has the following meaning:

- If $x_j > 0$, then the amount of asset j is increased (by purchasing).
- If $x_j < 0$, then the amount of asset j is decreased (by selling).

The model deals with two central quantities:

- Expected return:
$$E[r^T(w^0 + x)] = \bar{r}^T(w^0 + x).$$
- Variance (Risk):
$$V[r^T(w^0 + x)] = (w^0 + x)^T \Sigma (w^0 + x).$$

By definition Σ is positive semi-definite and

$$\begin{array}{rcl} \text{Std. dev.} & = & \left\| \Sigma^{\frac{1}{2}}(w^0 + x) \right\| \\ & = & \left\| L^T(w^0 + x) \right\| \end{array}$$

where L is **any** matrix such that

$$\Sigma = LL^T$$

A low rank of Σ is advantageous from a computational point of view. A valid L can always be computed as the Cholesky factorization of Σ.

9.7.2.1 Minimizing variance for a given return

In our first model we want to minimize the variance while selecting a portfolio with a specified expected target return t. Additionally the portfolio must satisfy the budget (self-financing) constraint asserting that the total amount of assets sold must equal the total amount of assets purchased. This is expressed in the model

$$\begin{array}{ll} \text{minimize} & V[r^T(w^0+x)] \\ \text{subject to} & E[r^T(w^0+x)] \;=\; t, \\ & e^T x \;=\; 0, \end{array} \tag{9.36}$$

where $e := (1,\ldots,1)^T$. Using the definitions above this may be formulated as a quadratic optimization problem:

$$\begin{array}{ll} \text{minimize} & (w^0+x)^T \Sigma (w^0+x) \\ \text{subject to} & \bar{r}^T(w^0+x) \;=\; t, \\ & e^T x \;=\; 0, \end{array} \tag{9.37}$$

9.7.2.2 Conic quadratic reformulation.

An equivalent conic quadratic reformulation is given by:

$$\begin{array}{lrcl} \text{minimize} & f & & \\ \text{subject to} & \Sigma^{\frac{1}{2}}(w^0+x)-g & = & 0, \\ & \bar{r}^T(w^0+x) & = & t, \\ & e^T x & = & 0, \\ & f \geq \|g\| \,. & & \end{array} \tag{9.38}$$

Here we minimize the standard deviation instead of the variance. Please note that $\Sigma^{\frac{1}{2}}$ can be replaced by any matrix L where $\Sigma = LL^T$. A low rank L is computationally advantageous.

9.7.2.3 Transaction costs with market impact term

We will now expand our model to include transaction costs as a fraction of the traded volume. [1, pp. 445-475] argues that transaction costs can be modelled as follows

$$\text{commission} + \frac{\text{bid}}{\text{ask}} - \text{spread} + \theta\sqrt{\frac{\text{trade volume}}{\text{daily volume}}}, \tag{9.39}$$

and that these are important to incorporate into the model.

In the following we deal with the last of these terms denoted the *market impact term*. If you sell (buy) a lot of assets the price is likely to go down (up). This can be captured in the market impact term

$$\theta\sqrt{\frac{\text{trade volume}}{\text{daily volume}}} \approx m_j\sqrt{|x_j|}.$$

The θ and "daily volume" have to be estimated in some way, i.e.

$$m_j = \frac{\theta}{\sqrt{\text{daily volume}}}$$

has to be estimated. The market impact term gives the cost as a fraction of daily traded volume ($|x_j|$). Therefore, the total cost when trading an amount x_j of asset j is given by

$$|x_j|(m_j|x_j|^{\frac{1}{2}}).$$

This leads us to the model:

$$\begin{array}{lrcl} \text{minimize} & f & & \\ \text{subject to} & \Sigma^{\frac{1}{2}}(w^0+x)-g & = & 0, \\ & \bar{r}^T(w^0+x) & = & t, \\ & e^Tx+e^Ty & = & 0, \\ & |x_j|(m_j|x_j|^{\frac{1}{2}}) & \leq & y_j, \\ & f \geq \|g\|. & & \end{array} \tag{9.40}$$

Now, defining the variable transformation

$$y_j = m_j\bar{y}_j$$

we obtain

$$\begin{array}{lrcl} \text{minimize} & f & & \\ \text{subject to} & \Sigma^{\frac{1}{2}}(w^0+x)-g & = & 0, \\ & \bar{r}^T(w^0+x) & = & t, \\ & e^Tx+m^T\bar{y} & = & 0, \\ & |x_j|^{3/2} & \leq & \bar{y}_j, \\ & f \geq \|g\|. & & \end{array} \tag{9.41}$$

As shown in Section 9.4.4.3 the set

$$|x_j|^{3/2} \leq \bar{y}_j$$

can be modelled by

$$\begin{array}{rcl} x_j & \leq & z_j, \\ -x_j & \leq & z_j, \\ z_j^2 & \leq & 2s_j\bar{y}_j, \\ u_j^2 & \leq & 2v_jq_j, \\ z_j & = & v_j, \\ s_j & = & u_j, \\ q_j & = & \frac{1}{8}, \\ q_j, s_j, \bar{y}_j, v_j, q_j & \geq & 0. \end{array} \tag{9.42}$$

9.7.2.4 Further reading

For further reading please see the reader to [21] in particular, and [24] and [1], which also contain relevant material.

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: Solving the actual optimization.

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 normally preprocessed using the so-called presolve. The purpose of the presolve is to

- remove redundant constraints,

- eliminate fixed variables,
- remove linear dependencies,
- substitute out free variables, and
- reduce the size of the optimization problem in general.

After the presolved problem has been optimized the solution is automatically postsolved so that the returned solution is valid for the original problem. Hence, the presolve is completely transparent. For further details about the presolve phase, please see [4, 6].

It is possible to fine-tune the behavior of the presolve or to turn it off entirely. If the presolve is known to be unable to reduce the size of a problem significantly, then turning off the presolve is beneficial. 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 usually

- the eliminator, and
- the linear dependency check.

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

The purpose of the eliminator is to eliminate free and implied free variables from the problem using substitution. For instance, given the constraints

$$\begin{array}{rcl} y & = & \sum_j x_j, \\ y, x & \geq & 0, \end{array}$$

y is an implied free variable that can be substituted out of the problem, if deemed worthwhile. By implied free variable is meant that the constraint $y \geq 0$ is redundant and hence y can be treated as a free variable.

For large scale problems the eliminator usually removes many constraints and variables. However, in some cases few or no eliminations can be performed and moreover, the eliminator may consume a lot of memory and time. If this is the case it is worthwhile to disable the eliminator by setting the parameter `MSK_IPAR_PRESOLVE_ELIMINATOR_USE` to `MSK_OFF`.

The purpose of the linear dependency check is to remove linear dependencies among the linear equalities. For instance, the three linear equalities

$$\begin{array}{rcl} x_1 + x_2 + x_3 & = & 1, \\ x_1 + 0.5x_2 & = & 0.5, \\ 0.5x_2 + x_3 & = & 0.5 \end{array}$$

contain exactly one linear dependency. This implies that one of the constraints can be dropped without changing the set of feasible solutions, i.e. one of the constraints is redundant. 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, and therefore it is strongly recommended to build models without linear dependencies. In case the linear dependencies are removed at the modelling stage, the linear dependency check can safely be disabled by setting the parameter `MSK_IPAR_PRESOLVE_LINDEP_USE` to `MSK_OFF`.

10.1.2 Dualizer

It is well-known that all linear, conic, and convex optimization problems have an associated dual problem. Moreover, even if the dual instead of the primal problem is solved, it is possible to recover the solution to the original primal problem.

In general, it is very hard to say whether it is easier to solve the primal or the dual problem but MOSEK has some heuristics for deciding which of the two problems to solve. Which form of the problem (primal or dual) that is solved is displayed in the MOSEK log. Please note that the dualizer is transparent, and all solution values returned by the optimizer refer to the original primal problem.

The dualizer can be controlled 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.

Finally, please 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 might 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 CPU's

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

By default MOSEK uses one thread to solve the problem, but the number of threads (and thereby CPUs) employed can be changed by setting the parameter `MSK_IPAR_INTPNT_NUM_THREADS` This should never exceed the number of CPU's on the machine.

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

For small problems, using multiple threads will probably not be worthwhile.

10.2 Linear optimization

10.2.1 Optimizer selection

For linear optimization problems two different types of optimizers are available. The default for linear problems is an interior-point optimizer, however, as an alternative the simplex optimizer can be employed.

The curious reader can consult [25] for a discussion about interior-point and simplex algorithms.

10.2.2 The interior-point optimizer

The MOSEK interior-point optimizer is an implementation of the homogeneous and self-dual algorithm. For a detailed description of the algorithm, please see [11].

10.2.2.1 Basis identification

It is well-known that 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 is found in [8].

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

Parameter name	Purpose
MSK_DPAR_INTPNT_TOL_PFEAS	Controls primal feasibility.
MSK_DPAR_INTPNT_TOL_DFEAS	Controls dual feasibility.
MSK_DPAR_INTPNT_TOL_REL_GAP	Controls relative gap.
MSK_DPAR_INTPNT_TOL_INFEAS	Controls when the problem is declared primal or dual infeasible.
MSK_DPAR_INTPNT_TOL_MU_RED	Controls when the complementarity is reduced enough.

Table 10.1: Parameters employed in termination criterion.

- MSK_IPAR_BI_IGNORE_NUM_ERROR

controls when basis identification is performed.

10.2.2.2 Interior-point termination criterion

The parameters in Table 10.1 control when the interior-point optimizer terminates.

10.2.3 The simplex based optimizer

An alternative to the interior-point optimizer is the simplex optimizer. The simplex optimizer employs a different approach than the interior-point optimizer when solving a problem. Contrary to the interior-point optimizer the simplex optimizer can exploit a guess for the optimal solution to reduce solution time. Depending on the problem it may be faster or slower to exploit a guess for the optimal solution. 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 (9.1) and (9.2) 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 given the 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

MOSEK is designed to minimize numerical difficulties, however, in rate cases the optimizer may have a hard time solving a problem. MOSEK counts a numerical unexpected behavior inside the optimizer as a "set-back". The user can define how many set-backs the optimizer accepts, and 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.

What counts as a set-back? It is hard to say without getting very technical but obvious cases are repeated singularities when factorizing the basis matrix, repeated loss of feasibility, degeneracy problems (no progress in objective) or 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. If a lot of set-backs still occur, then 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 simply impossible, however, in recent years the dual optimizer has experienced 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 the network flow structure specified in Section 9.2 can in most cases be solved significantly faster with a specialized version of the simplex method [2], rather than with the general solvers.

MOSEK includes a network simplex solver, which usually solves network problems 10 to 100 times faster than the standard simplex optimizers implemented by MOSEK.

To use the network simplex optimizer, do the following

- Input the network flow problem as an ordinary linear optimization problem.
- Set
  - the `MSK_IPAR_SIM_NETWORK_DETECT` parameter to 0, and
  - the `MSK_IPAR_OPTIMIZER` parameter to `MSK_OPTIMIZER_FREE_SIMPLEX`.
- Optimize the problem.

MOSEK will automatically detect the network structure and apply the specialized simplex optimizer.

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.

10.3.2 Embedded network problems

Often problems contains both large parts with network structure and some non-network constraints or variables — such problems are said to have *embedded network structure*. If the procedure described above is applied, MOSEK will try to exploit this structure to speed up the optimization.

This is done by heuristically detecting the largest network embedded in the problem, solving this using the network simplex optimizer, and using this solution to hot-start a normal simplex optimizer.

The `MSK_IPAR_SIM_NETWORK_DETECT` parameter defines how large a percentage of the problem should be a network before the specialized solver is applied. In general, it is recommended to use the network optimizer only on problems containing a substantial embedded network.

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 [7].

10.4.1.1 Interior-point termination criteria

The parameters controlling when the conic interior-point optimizer terminates are shown in Table 10.2.

10.5 Nonlinear convex optimization

10.5.1 The interior-point optimizer

For quadratic, quadratically constrained, and general convex optimization problems only an interior-point type optimizer is available. The interior-point optimizer is an implementation of

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.

the homogeneous and self-dual algorithm. For a detailed description of the algorithm, please see [9, 10].

10.5.1.1 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 CPU gets a share of the work. The user decides how many CPUs MOSEK should exploit.

It is not always possible to divide the work equally, and often parts of the computations and the coordination of the work is processed sequentially, even if several CPUs are present. Therefore, the speed-up obtained when using multiple CPUs is highly problem dependent. However, as a rule of thumb, if the problem solves very quickly, i.e. in less than 60 seconds, it is not advantageous to use the parallel option.

The MSK_IPAR_INTPNT_NUM_THREADS parameter sets the number of threads (and therefore the number of CPUs) that the interior point optimizer will use.

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.

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

```
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.

10.6.4 A more flexible concurrent optimizer

MOSEK also provides a more flexible method of concurrent optimization by using the function `MSK_optimizeconcurrent`. The main advantages of this function are that it allows the calling application to assign arbitrary values to the parameters of each tasks, and that call-back functions can be attached to each task. This may be useful in the following situation: Assume that you know the primal simplex optimizer to be the best optimizer for your problem, but that you do not know which of the available selection strategies (as defined by the `MSK_IPAR_SIM_PRIMAL_SELECTION` parameter) is the best. In this case you can solve the problem with the primal simplex optimizer using several different selection strategies concurrently.

An example demonstrating the usage of the `MSK_optimizeconcurrent` function is included below. The example solves a single problem using the interior-point and primal simplex optimizers in parallel.

Chapter 11

The optimizer for mixed integer problems

A problem is a mixed integer optimization problem when one or more of the variables are constrained to be integers. The integer optimizer available in MOSEK can solve integer optimization problems involving

- linear,
- quadratic and
- conic

constraints. However, a problem is not allowed to have both conic constraints and quadratic objective or constraints.

Readers unfamiliar with integer optimization are strongly recommended to consult some relevant literature, e.g. the book [28] by Wolsey is a good introduction to integer optimization.

11.1 Some notation

In general, an integer optimization problem has the form

$$\begin{array}{rclccccl} z^* & = & \text{minimize} & & & c^T x & & \\ & & \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & & & l^x & \leq & Ax & \leq & u^x, \\ & & & & & x_j \in \mathcal{Z}, & & \forall j \in \mathcal{J}, \end{array} \tag{11.1}$$

where $\mathcal{J}$ is an index set specifying which variables are integer constrained. Frequently we talk about the continuous relaxation of an integer optimization problem defined as

$$\begin{array}{rclccccl} \underline{z} & = & \text{minimize} & & & c^T x & & \\ & & \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & & & l^x & \leq & Ax & \leq & u^x \end{array} \tag{11.2}$$

i.e. we ignore the constraint

$$x_j \in \mathcal{Z}, \ \forall j \in \mathcal{J}.$$

Moreover, let $\hat{x}$ be any feasible solution to (11.1) and define

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

It should be obvious that

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

holds. This is an important observation since if we assume that it is not possible to solve the mixed integer optimization problem within a reasonable time frame, but that a feasible solution can be found, then the natural question is: How far is the *obtained* solution from the *optimal* solution? The answer is that no feasible solution can have an objective value smaller than $\underline{z}$, which implies that the obtained solution is no further away from the optimum than $\overline{z} - \underline{z}$.

11.2 An important fact about integer optimization problems

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

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

11.3 How the integer optimizer works

The process of solving an integer optimization problem can be split in three phases:

Presolve: In this phase the optimizer tries to reduce the size of the problem using preprocessing techniques. Moreover, it strengthens the continuous relaxation, if possible.

Heuristic: Using heuristics the optimizer tries to guess a good feasible solution.

Optimization: The optimal solution is located using a variant of the branch-and-cut method.

In some cases the integer optimizer may locate an optimal solution in the preprocessing stage or conclude that the problem is infeasible. Therefore, the heuristic and optimization stages may never be performed.

11.3.1 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 .

11.3.2 Heuristic

Initially, the integer optimizer tries to guess a good feasible solution using different heuristics:

- First a very simple rounding heuristic is employed.
- Next, if deemed worthwhile, the *feasibility pump* heuristic is used.
- Finally, if the two previous stages did not produce a good initial solution, more sophisticated heuristics are used.

The following parameters can be used to control the effort made by the integer optimizer to find an initial feasible solution.

- `MSK_IPAR_MIO_HEURISTIC_LEVEL`: Controls how sophisticated and computationally expensive a heuristic to employ.
- `MSK_DPAR_MIO_HEURISTIC_TIME`: The minimum amount of time to spend in the heuristic search.
- `MSK_IPAR_MIO_FEASPUMP_LEVEL`: Controls how aggressively the feasibility pump heuristic is used.

11.3.3 The optimization phase

This phase solves the problem using the branch and cut algorithm.

11.4 Termination criterion

In general, it is impossible to find an exact feasible and optimal solution to an integer optimization problem in a reasonable amount of time, though in many practical cases it may be possible. Therefore, the integer optimizer employs a relaxed feasibility and optimality criterion to determine when a satisfactory solution is located.

A candidate solution, i.e. a solution to (11.2), is said to be an integer feasible solution if the criterion

$$\min(|x_j| - \lfloor x_j \rfloor, \lceil x_j \rceil - |x_j|) \leq \max(\delta_1, \delta_2 |x_j|)\ \forall j \in \mathcal{J}$$

is satisfied. Hence, such a solution is defined as a feasible solution to (11.1).

Tolerance	Parameter name
δ_1	MSK_DPAR_MIO_TOL_ABS_RELAX_INT
δ_2	MSK_DPAR_MIO_TOL_REL_RELAX_INT
δ_3	MSK_DPAR_MIO_TOL_ABS_GAP
δ_4	MSK_DPAR_MIO_TOL_REL_GAP
δ_5	MSK_DPAR_MIO_NEAR_TOL_ABS_GAP
δ_6	MSK_DPAR_MIO_NEAR_TOL_REL_GAP

Table 11.1: 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.

Table 11.2: Parameters affecting the termination of the integer optimizer.

Whenever the integer optimizer locates an integer feasible solution it will check if the criterion

$$\overline{z} - \underline{z} \leq \max(\delta_3, \delta_4 \max(1, |\overline{z}|))$$

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

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

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

The δ tolerances can are specified using parameters — see Table 11.1. If an optimal solution cannot be located within a reasonable time, it may be advantageous to employ a relaxed termination criterion after some time. Whenever the integer optimizer locates an integer feasible solution and has spent at least the number of seconds defined by the MSK_DPAR_MIO_DISABLE_TERM_TIME parameter on solving the problem, it will check whether the criterion

$$\overline{z} - \underline{z} \leq \max(\delta_5, \delta_6 \max(1, |\overline{z}|))$$

is satisfied. If it is satisfied, the optimizer will report that the candidate solution is **near optimal** and then terminate. All δ tolerances can be adjusted using suitable parameters — see Table 11.1. In Table 11.2 some other parameters affecting the integer optimizer termination criterion are shown. Please note that if the effect of a parameter is delayed, the associated termination criterion is applied only after some time, specified by the MSK_DPAR_MIO_DISABLE_TERM_TIME parameter.

11.5 How to speed up the solution process

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

- Relax the termination criterion: In case the run time is not acceptable, the first thing to do is to relax the termination criterion — see Section 11.4 for details.
- Specify a good initial solution: In many cases a good feasible solution is either known or easily computed using problem specific knowledge. If a good feasible solution is known, it is usually worthwhile to use this as a starting point for the integer optimizer.
- Improve the formulation: A mixed integer optimization problem may be impossible to solve in one form and quite easy in another form. However, it is beyond the scope of this manual to discuss good formulations for mixed integer problems. For discussions on this topic see for example [28].

Chapter 12

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.

12.1 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 12.1.

The problem represented in figure 12.1 is infeasible, since the total demand

$$2300 = 1100 + 200 + 500 + 500 \tag{12.1}$$

exceeds the total supply

$$2200 = 200 + 1000 + 1000 \tag{12.2}$$

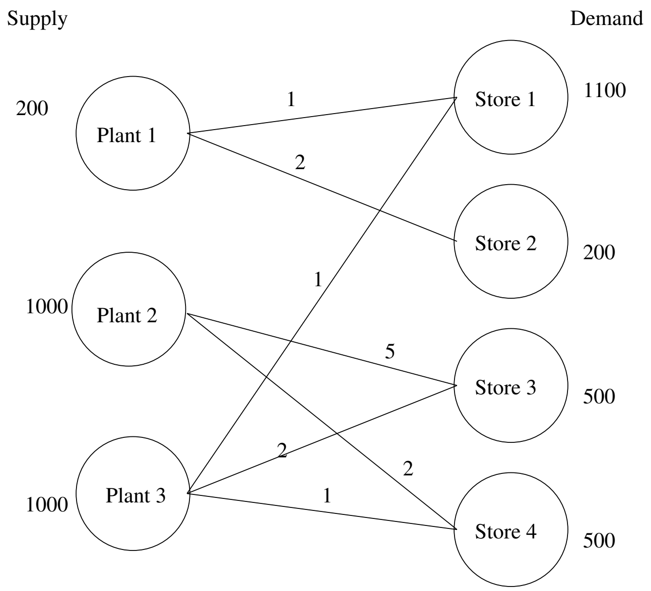

Figure 12.1: Supply, demand and cost of transportation.

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:

$$
\begin{array}{lrcrcrcrcrcrcrcl}
\text{minimize} & x_{11} & + & 2x_{12} & + & 5x_{23} & + & 2x_{24} & + & x_{31} & + & 2x_{33} & + & x_{34} & & \\
\text{subject to} & x_{11} & + & x_{12} & & & & & & & & & & & \leq & 200, \\
 & & & & & x_{23} & + & x_{24} & & & & & & & \leq & 1000, \\
 & & & & & & & & & x_{31} & + & x_{33} & + & x_{34} & \leq & 1000, \\
 & x_{11} & & & & & & & + & x_{31} & & & & & = & 1100, \\
 & & & x_{12} & & & & & & & & & & & = & 200, \\
 & & & & & x_{23} & + & & & & & x_{33} & & & = & 500, \\
 & & & & & & & x_{24} & + & & & & & x_{34} & = & 500, \\
 & x_{ij} \geq 0. & & & & & & & & & & & & & &
\end{array}
\tag{12.3}
$$

Solving the problem (12.3) using MOSEK will result in a solution, a solution status and a problem status. Among the log output from the execution of MOSEK on the above problem are the lines:

```
Basic solution
Problem status  : PRIMAL_INFEASIBLE
Solution status : PRIMAL_INFEASIBLE_CER
```

The first line indicates that the problem status is primal infeasible. The second line says that a *certificate of the infeasibility* was found. The certificate is returned in place of the solution to the problem.

12.1.1 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 12.1.3) may assist you in finding the constraints causing the infeasibility.

Possible ways of relaxing your problem include:

- Increasing (decreasing) upper (lower) bounds on variables and constraints.
- Removing suspected constraints from the problem.

Returning to the transportation example, we discover that removing the fifth constraint

$$x_{12} = 200 \tag{12.4}$$

makes the problem feasible.

12.1.2 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:

$$\begin{array}{ll} \text{minimize} & x_1 \\ \text{subject to} & x_1 \leq 5 \end{array} \tag{12.5}$$

To resolve a dual infeasibility the primal problem must be made more restricted by

- Adding upper or lower bounds on variables or constraints.
- Removing variables.
- Changing the objective.

12.1.2.1 A cautious note

The problem

$$\begin{array}{lll} \text{minimize} & 0 & \\ \text{subject to} & 0 \leq x_1, & \\ & x_j \leq x_{j+1}, & j = 1, \ldots, n-1, \\ & x_n \leq -1 & \end{array} \tag{12.6}$$

is clearly infeasible. Moreover, if any one of the constraints are dropped, then the problem becomes feasible.

This illustrates the worst case scenario that all, or at least a significant portion, of the constraints are involved in the infeasibility. Hence, it may not always be easy or possible to pinpoint a few constraints which are causing the infeasibility.

12.1.3 The infeasibility report

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

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

12.1.3.1 Example: Primal infeasibility

We will reuse the example (12.3) located in `infeas.lp`:

```
\
\ An example of an infeasible linear problem.
\
minimize
 obj: + 1 x11 + 2 x12 + 1 x13
      + 4 x21 + 2 x22 + 5 x23
      + 4 x31 + 1 x32 + 2 x33
st
  s0: + x11 + x12      <= 200
  s1: + x23 + x24      <= 1000
  s2: + x31 +x33 + x34 <= 1000
  d1: + x11 + x31       = 1100
  d2: + x12             = 200
  d3: + x23 + x33       = 500
  d4: + x24 + x34       = 500
bounds
end
```

Using the command line

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

MOSEK produces the following infeasibility report

```
MOSEK PRIMAL INFEASIBILITY REPORT.

Problem status: The problem is primal infeasible

The following constraints are involved in the primal infeasibility.

Index     Name       Lower bound       Upper bound       Dual lower        Dual upper
0         s0         NONE              2.000000e+002     0.000000e+000     1.000000e+000
2         s2         NONE              1.000000e+003     0.000000e+000     1.000000e+000
3         d1         1.100000e+003     1.100000e+003     1.000000e+000     0.000000e+000
4         d2         2.000000e+002     2.000000e+002     1.000000e+000     0.000000e+000

The following bound constraints are involved in the infeasibility.

Index     Name       Lower bound       Upper bound       Dual lower        Dual upper
8         x33        0.000000e+000     NONE              1.000000e+000     0.000000e+000
10        x34        0.000000e+000     NONE              1.000000e+000     0.000000e+000
```

The infeasibility report is divided into two sections where the first section shows which constraints that are important for the infeasibility. In this case the important constraints are the ones named `s0`, `s2`, `d1`, and `d2`. The values in the columns “`Dual lower`” and “`Dual upper`” are also useful,since a non-zero *dual lower* value for a constraint implies that the lower bound on the constraint is important for the infeasibility. Similarly, a non-zero *dual upper* value implies that the upper bound on the constraint is important for the infeasibility.

It is also possible to obtain the infeasible subproblem. The executing the command

```
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. Please note that this optimization problem is identical to (12.3), except that the objective and some of the constraints and bounds have been removed. Executing the command

```
mosek -d MSK_IPAR_INFEAS_REPORT_AUTO MSK_ON rinfeas.bas.inf.lp
```

demonstrates that the reduced problem is **primal infeasible**. However, since the reduced problem is usually smaller, it should be easier to locate the cause of the infeasibility in this rather than in the original problem (12.3).

12.1.3.2 Example: Dual infeasibility

The example problem

```
minimize -  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):

```
he 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

$$c^t x^* > 0. \tag{12.7}$$

For a minimization problem this inequality would have been reversed — see (12.19).

From the infeasibility report we see that the variable `y4`, and the constraints `x11` and `x33` are involved in the infeasibility since these appear with non-zero values in the "`Activity`" column.

One possible strategy to "fix" the infeasibility is to modify the problem so that the certificate of infeasibility becomes invalid. In this case we might do one the the following things:

- Put a lower bound in `y3`. This will directly invalidate the certificate of dual infeasibility.
- Increase the object coefficient of `y3`. Changing the coefficients sufficiently will invalidate the inequality (12.7) and thus the certificate.
- Put lower bounds on `x11` or `x31`. This will directly invalidate the certificate of infeasibility.

Please note that modifying the problem to invalidate the reported certificate does *not* imply that the problem becomes dual feasible — the infeasibility may simply "move", resulting in a new infeasibility.

More often, the reported certificate can be used to give a hint about errors or inconsistencies in the model that produced the problem.

12.2 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

$$\begin{array}{lcccl} \text{minimize} & & c^T x + c^f & & \\ \text{subject to} \; l^c & \leq & Ax & \leq & u^c, \\ l^x & \leq & x & \leq & u^x \end{array} \tag{12.8}$$

where the corresponding dual problem is

$$\begin{array}{lll} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c & \\ & +(l^x)^T s_l^x - (u^x)^T s_u^x + c^f & \\ \text{subject to} & A^T y + s_l^x - s_u^x & = \; c, \\ & -y + s_l^c - s_u^c & = \; 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0. & \end{array} \tag{12.9}$$

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

$$l_j^x = -\infty \;\Rightarrow\; (s_l^x)_j = 0 \tag{12.10}$$

12.2.1 Certificat of primal infeasibility

A certificate of primal infeasibility is *any* solution to the homogenized dual problem

$$\begin{array}{lll} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c & \\ & +(l^x)^T s_l^x - (u^x)^T s_u^x & \\ \text{subject to} & A^T y + s_l^x - s_u^x & = \; 0, \\ & -y + s_l^c - s_u^c & = \; 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0. & \end{array} \tag{12.11}$$

with a positive objective value. That is, $(s_l^{c*}, s_u^{c*}, s_l^{x*}, s_u^{x*})$ is a certificat of primal infeasibility if

$$(l^c)^T s_l^{c*} - (u^c)^T s_u^{c*} + (l^x)^T s_l^{x*} - (u^x)^T s_u^{x*} > 0 \tag{12.12}$$

and

$$\begin{array}{lcl} A^T y + s_l^{x*} - s_u^{x*} & = & 0, \\ -y + s_l^{c*} - s_u^{c*} & = & 0, \\ s_l^{c*}, s_u^{c*}, s_l^{x*}, s_u^{x*} \geq 0. & & \end{array} \tag{12.13}$$

The well-known Farkas Lemma tells us that (12.8) is infeasible if and only if a certificat of primal infeasibility exists.

Let $(s_l^{c*}, s_u^{c*}, s_l^{x*}, s_u^{x*})$ be a certificate of primal infeasibility then

$$(s_l^{c*})_i > 0 \quad ((s_u^{c*})_i > 0) \tag{12.14}$$

implies that the lower (upper) bound on the ith constraint is important for the infeasibility. Furthermore,

$$(s_l^{x*})_j > 0 \quad ((s_u^{x*})_i > 0) \tag{12.15}$$

implies that the lower (upper) bound on the jth variable is important for the infeasibility.

12.2.2 Certificat of dual infeasibility

A certificate of dual infeasibility is *any* solution to the problem

$$\begin{array}{lccccc} \text{minimize} & & & c^T x & & \\ \text{subject to} & \bar{l}^c & \leq & Ax & \leq & \bar{u}^c, \\ & \bar{l}^x & \leq & x & \leq & \bar{u}^x \end{array} \tag{12.16}$$

with negative objective value, where we use the definitions

$$\bar{l}_i^c := \left\{ \begin{array}{ll} 0, & l_i^c > -\infty, \\ -\infty, & \text{otherwise}, \end{array} \right. \qquad \bar{u}_i^c := \left\{ \begin{array}{ll} 0, & u_i^c < \infty, \\ \infty, & \text{otherwise}, \end{array} \right. \tag{12.17}$$

and

$$\bar{l}_i^x := \left\{ \begin{array}{ll} 0, & l_i^x > -\infty, \\ -\infty, & \text{otherwise}, \end{array} \right. \text{ and } \bar{u}_i^x := \left\{ \begin{array}{ll} 0, & u_i^x < \infty, \\ \infty, & \text{otherwise}. \end{array} \right. \tag{12.18}$$

Stated differently, a certificate of dual infeasibility is any x^* such that

$$\begin{array}{ccccc} & & c^T x^* & < & 0, \\ \bar{l}^c & \leq & Ax^* & \leq & \bar{u}^c, \\ \bar{l}^x & \leq & x^* & \leq & \bar{u}^x \end{array} \tag{12.19}$$

The well-known Farkas Lemma tells us that (12.9) is infeasible if and only if a certificat of dual infeasibility exists.

Observe that if x^* is a certificate of dual infeasibility then for any j such that

$$x_j^* \neq 0, \tag{12.20}$$

variable j is involved in the dual infeasibility.

Chapter 13

Sensitivity analysis

13.1 Introduction

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

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

13.2 Restrictions

Currently, sensitivity analysis is only available for continuous linear optimization problems. Moreover, MOSEK can only deal with perturbations in bounds or objective coefficients.

13.3 References

The book [17] discusses the classical sensitivity analysis in Chapter 10 whereas the book [23, Chapter 19] presents a modern introduction to sensitivity analysis. Finally, it is recommended to read the short paper [26] to avoid some of the pitfalls associated with sensitivity analysis.

13.4 Sensitivity analysis for linear problems

13.4.1 The optimal objective value function

Assume we are given the problem

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

and we want to know how the optimal objective value changes as l^c_i is perturbed. In order to answer this question then define the perturbed problem for l^c_i as follows

$$\begin{array}{rclcccc} f_{l^c_i}(\beta) & = & \text{minimize} & & c^T x & & \\ & & \text{subject to} & l^c + \beta e_i \leq & Ax & \leq u^c, & \\ & & & & l^x \leq x \leq u^x, & & \end{array} \tag{13.2}$$

where e_i is the ith column of the identity matrix. The function

$$f_{l^c_i}(\beta) \tag{13.3}$$

shows the optimal objective value as a function of β. Note a change in β corresponds to a perturbation in l^c_i and hence (13.3) shows the optimal objective value as a function of l^c_i.

It is possible to prove that the function (13.3) is a piecewise linear and convex function i.e. the function may look like the illustration in Figure 13.1.

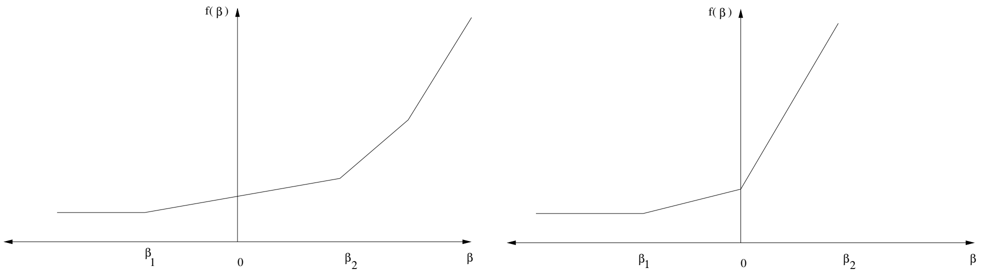

Figure 13.1: The optimal value function $f_{l^c_i}(\beta)$. Left: $\beta = 0$ is in the interior of linearity interval. Right: $\beta = 0$ is a breakpoint.

Clearly, if the function $f_{l^c_i}(\beta)$ does not change much when β is changed, then we can conclude that the optimal objective value is insensitive to changes in l^c_i. Therefore, we are interested in how $f_{l^c_i}(\beta)$ changes for small changes in β. Now define

$$f'_{l^c_i}(0) \tag{13.4}$$

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

$$\beta \in [\beta_1, \beta_2] \tag{13.5}$$

for which

$$f'_{l_i^c}(\beta) = f'_{l_i^c}(0). \tag{13.6}$$

To summarize the sensitivity analysis provides a shadow price and the linearity interval in which the shadow price is constant.

The reader may have noticed that we are sloppy in the definition of the shadow price. The reason is that the shadow price is not defined in the right example in Figure 13.1 because the function $f_{l_i^c}(\beta)$ is not differentiable for $\beta = 0$. However, in that case we can define a left and a right shadow price and a left and a right linearity interval.

In the above discussion we only discussed changes in l_i^c. We define the other optimal objective value functions as follows

$$\begin{array}{rcll} f_{u_i^c}(\beta) & = & z(l^c, u^c + \beta e_i, l^x, u^x, c), & i = 1, \ldots, m, \\ f_{l_j^x}(\beta) & = & z(l^c, u^c, l^x + \beta e_j, u^x, c), & j = 1, \ldots, n, \\ f_{u_j^x}(\beta) & = & z(l^c, u^c, l^x, u^x + \beta e_j, c), & j = 1, \ldots, n, \\ f_{c_j}(\beta) & = & z(l^c, u^c, l^x, u^x, c + \beta e_j), & j = 1, \ldots, n. \end{array} \tag{13.7}$$

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

13.4.1.1 Equality constraints

In MOSEK a constraint can be specified as either an equality constraints or a ranged constraints. Suppose constraint i is an equality constraint. We then define the optimal value function for constraint i by

$$f_{e_i^c}(\beta) = z(l^c + \beta e_i, u^c + \beta e_i, l^x, u^x, c) \tag{13.8}$$

Thus for a equality constraint the upper and lower bound (which are equal) are perturbed simultaneously. From the point of view of MOSEK sensitivity analysis a ranged constrain with $l_i^c = u_i^c$ therefore differs from an equality constraint.

13.4.2 The basis type sensitivity analysis

The classical sensitivity analysis discussed in most textbooks about linear optimization, e.g. [17, Chapter 10], is based on an optimal basic solution or equivalently on an optimal basis. This method may produce misleading results [23, Chapter 19] but is **computationally cheap**. Therefore, and for historical reasons this method is available in MOSEK.

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

13.4.3 The optimal partition type sensitivity analysis

Another method for computing the complete linearity interval is called the *optimal partition type sensitivity analysis*. The main drawback to the optimal partition type sensitivity analysis is it is computationally expensive. This type of sensitivity analysis is currently provided as an experimental feature in MOSEK.

Given optimal primal and dual solutions to (13.1) i.e. x^* and $((s_l^c)^*, (s_u^c)^*, (s_l^x)^*, (s_u^x)^*)$ then the optimal objective value is given by

$$z^* := c^T x^*. \tag{13.9}$$

The left and right shadow prices σ_1 and σ_2 for l_i^c is given by the pair of optimization problems

$$\begin{array}{rclccl} \sigma_1 & = & \text{minimize} & e_i^T s_l^c & & \\ & & \text{subject to} & A^T(s_l^c - s_u^c) + s_l^x - s_u^x & = & c, \\ & & & (l_c)^T(s_l^c) - (u_c)^T(s_u^c) + (l_x)^T(s_l^x) - (u_x)^T(s_u^x) & = & z^*, \\ & & & s_l^c, s_u^c, s_l^c, s_u^x \geq 0 & & \end{array} \tag{13.10}$$

and

$$\begin{array}{rclccl} \sigma_2 & = & \text{maximize} & e_i^T s_l^c & & \\ & & \text{subject to} & A^T(s_l^c - s_u^c) + s_l^x - s_u^x & = & c, \\ & & & (l_c)^T(s_l^c) - (u_c)^T(s_u^c) + (l_x)^T(s_l^x) - (u_x)^T(s_u^x) & = & z^*, \\ & & & s_l^c, s_u^c, s_l^c, s_u^x \geq 0. & & \end{array} \tag{13.11}$$

The above two optimization problems makes it easy to interpret-ate the shadow price. Indeed assume that $((s_l^c)^*, (s_u^c)^*, (s_l^x)^*, (s_u^x)^*)$ is an arbitrary optimal solution then it must hold

$$(s_l^c)_i^* \in [\sigma_1, \sigma_2]. \tag{13.12}$$

Next the linearity interval $[\beta_1, \beta_2]$ for l_i^c is computed by solving the two optimization problems

$$\begin{array}{rclrcl} \beta_1 & = & \text{minimize} & & \beta & \\ & & \text{subject to} & l^c + \beta e_i \leq & Ax & \leq u^c, \\ & & & & c^T x - \sigma_1 \beta & = z^*, \\ & & & & l^x \leq x \leq u^x, & \end{array} \tag{13.13}$$

and

$$\begin{array}{rclrcl} \beta_2 & = & \text{maximize} & & \beta & \\ & & \text{subject to} & l^c + \beta e_i \leq & Ax & \leq u^c, \\ & & & & c^T x - \sigma_2 \beta & = z^*, \\ & & & & l^x \leq x \leq u^x. & \end{array} \tag{13.14}$$

The linearity intervals and shadow prices for u_i^c, l_j^x, and u_j^x can be computed in a similar way to how it is computed for l_i^c.

The left and right shadow price for c_j denoted σ_1 and σ_2 respectively is given by the pair optimization problems

$$\begin{array}{rclrcl} \sigma_1 & = & \text{minimize} & & e_j^T x & \\ & & \text{subject to} & l^c + \beta e_i \leq & Ax & \leq u^c, \\ & & & & c^T x & = z^*, \\ & & & & l^x \leq x \leq u^x & \end{array} \tag{13.15}$$

and

$$\begin{array}{rclrcl} \sigma_2 & = & \text{maximize} & & e_j^T x & \\ & & \text{subject to} & l^c + \beta e_i \leq & Ax & \leq u^c, \\ & & & & c^T x & = z^*, \\ & & & & l^x \leq x \leq u^x. & \end{array} \tag{13.16}$$

Once again the above two optimization problems makes it easy to interpret-ate the shadow prices. Indeed assume that x^* is an arbitrary primal optimal solution then it must hold

$$x_j^* \in [\sigma_1, \sigma_2]. \tag{13.17}$$

The linearity interval $[\beta_1, \beta_2]$ for a c_j is computed as follows

$$\begin{array}{rcllcl} \beta_1 & = & \text{minimize} & \beta & & \\ & & \text{subject to} & A^T(s_l^c - s_u^c) + s_l^x - s_u^x & = & c + \beta e_j, \\ & & & (l_c)^T(s_l^c) - (u_c)^T(s_u^c) + (l_x)^T(s_l^x) - (u_x)^T(s_u^x) - \sigma_1 \beta & \leq & z^*, \\ & & & s_l^c, s_u^c, s_l^c, s_u^x \geq 0 & & \end{array} \tag{13.18}$$

and

$$\begin{array}{rcllcl} \beta_2 & = & \text{maximize} & \beta & & \\ & & \text{subject to} & A^T(s_l^c - s_u^c) + s_l^x - s_u^x & = & c + \beta e_j, \\ & & & (l_c)^T(s_l^c) - (u_c)^T(s_u^c) + (l_x)^T(s_l^x) - (u_x)^T(s_u^x) - \sigma_2 \beta & \leq & z^*, \\ & & & s_l^c, s_u^c, s_l^c, s_u^x \geq 0. & & \end{array} \tag{13.19}$$

13.4.4 An example

As an example we will use the following transportation problem. Consider the problem of minimizing the transportation cost between a number of production plants and stores. Each plant supplies a number of goods and each store has a given demand that must be met. Supply, demand and cost of transportation per unit are shown in Figure 13.2.

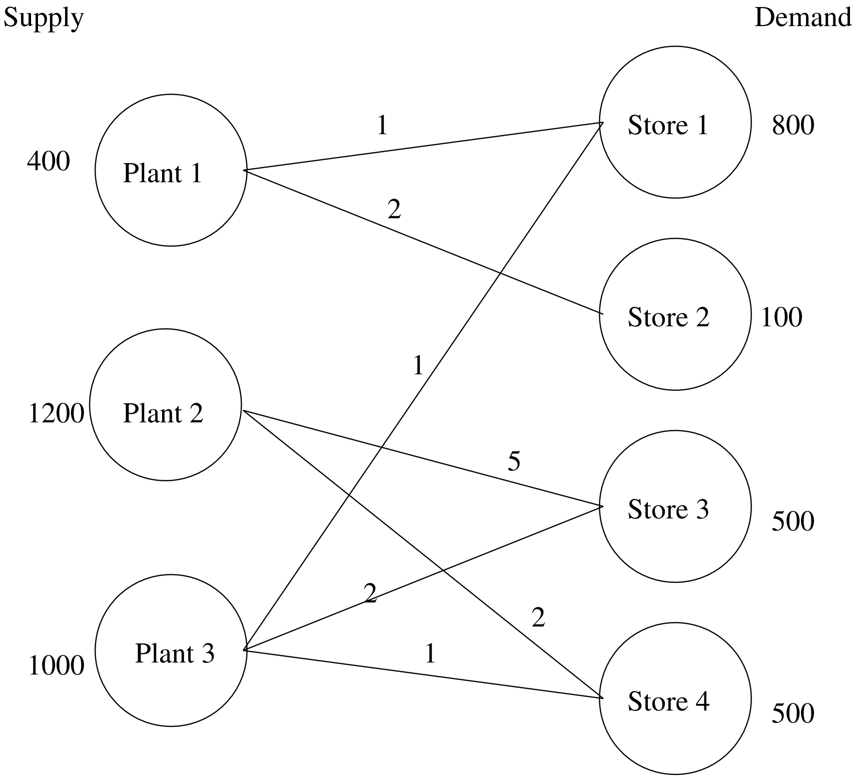

Figure 13.2: Supply, demand and cost of transportation.

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

minimize

$$1x_{11} + 2x_{12} + 5x_{23} + 2x_{24} + 1x_{31} + 2x_{33} + 1x_{34} \tag{13.20}$$

subject to

$$\begin{array}{cccccccccccccc}
x_{11} & + & x_{12} & & & & & & & & & & \le & 400, \\
 & & & x_{23} & + & x_{24} & & & & & & & \le & 1200, \\
 & & & & & & & x_{31} & + & x_{33} & + & x_{34} & \le & 1000, \\
x_{11} & & & & & & + & x_{31} & & & & & = & 800, \\
 & & x_{12} & & & & & & & & & & = & 100, \\
 & & & x_{23} & + & & & & & x_{33} & & & = & 500, \\
 & & & & & x_{24} & + & & & & & x_{34} & = & 500, \\
x_{11}, & & x_{12}, & x_{23}, & & x_{24}, & & x_{31}, & & x_{33}, & & x_{34} & \ge & 0.
\end{array} \tag{13.21}$$

The basis type and the optimal partition type sensitivity results for the transportation problem is shown in Table 13.1 and 13.2 respectively.

Basis type

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

Optimal partition type

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

Table 13.1: Ranges and shadow prices related to bounds on constraints and variables. Left: Results for basis type sensitivity analysis. Right: Results for the optimal partition type sensitivity analysis.

Looking at the results from the optimal partition type sensitivity analysis we see that for the constraint number 1 we have $\sigma_1 \neq \sigma_2$ and $\beta_1 \neq \beta_2$. Therefore, we have a left linearity interval of $[-300, 0]$ and a right interval of $[0, 500]$. The corresponding left and right shadow price is 3 and 1 respectively. This implies if the upper bound on constraint 1 increases by

$$\beta \in [0, \beta_1] = [0, 500] \tag{13.22}$$

Basis type

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

Optimal partition type

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

Table 13.2: Ranges and shadow prices related to the objective coefficients. Left: Results for basis type sensitivity analysis. Right: Results for the optimal partition type sensitivity analysis.

then the optimal objective value will decrease by the value

$$\sigma_2\beta = 1\beta. \tag{13.23}$$

Correspondingly, if the upper bound on constraint 1 is decreased by

$$\beta \in [0, 300] \tag{13.24}$$

then the optimal objective value will increased by the value

$$\sigma_1\beta = 3\beta. \tag{13.25}$$

13.5 Sensitivity analysis in the MATLAB toolbox

The following describe sensitivity analysis from the MATLAB toolbox.

13.5.1 On bounds

The index of bounds/variables to analyzed for sensitivity are specified in the following subfields of the matlab structure `prob`:

`.prisen.cons.subu`	Indexes of constraints, where upper bounds are analyzed for sensitivity.
`.prisen.cons.subl`	Indexes of constraints, where lower bounds are analyzed for sensitivity.
`.prisen.vars.subu`	Indexes of variables, where upper bounds are analyzed for sensitivity.
`.prisen.vars.subl`	Indexes of variables, where lower bounds are analyzed for sensitivity.

`.duasen.sub`	Index of variables where coefficients are analysed for sensitivity.

For an equality constraint, the index can be specified in either `subu` or `subl`. After calling

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

the results are returned in the subfields `prisen` and `duasen` of `res`.

13.5.1.1 prisen

The field `prisen` is structured as follows:

`.cons`	MATLAB structure with subfields:

`.lr_bl`	Left value β_1 in the linearity interval for a lower bound.
`.rr_bl`	Right value β_2 in the linearity interval for a lower bound.
`.ls_bl`	Left shadow price s_l for a lower bound.
`.rs_bl`	Right shadow price s_r for a lower bound.
`.lr_bu`	Left value β_1 in the linearity interval for an upper bound.
`.rr_bu`	Right value β_2 in the linearity interval for an upper bound.
`.ls_bu`	Left shadow price s_l for an upper bound.
`.rs_bu`	Right shadow price s_r for an upper bound.

`.var`	MATLAB structure with subfields:

`.lr_bl`	Left value β_1 in the linearity interval for a lower bound on a varable.
`.rr_bl`	Right value β_2 in the linearity interval for a lower bound on a varable.
`.ls_bl`	Left shadow price s_l for a lower bound on a varable.
`.rs_bl`	Right shadow price s_r for lower bound on a varable.
`.lr_bu`	Left value β_1 in the linearity interval for an upper bound on a varable.
`.rr_bu`	Right value β_2 in the linearity interval for an upper bound on a varable.
`.ls_bu`	Left shadow price s_l for an upper bound on a varables.
`.rs_bu`	Right shadow price s_r for an upper bound on a varables.

13.5.1.2 duasen

The field `duasen` is structured as follows:

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

13.5.2 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.

13.5.3 An example

Consider the problem defined in (13.21). Suppose we wish to perform sensitivity analysis on all bounds and coefficients. The following example demonstrates this as well as the method for changing between basic and full sensitivity analysis.

```
% sensitivity.m

% Obtain all symbolic constants
% defined by MOSEK.
[r,res]  = mosekopt('symbcon');
sc       = res.symbcon;
[r,res] = mosekopt('read(transport.lp) echo(0)');
prob = res.prob;
% analyse upper bound 1:7
prob.prisen.cons.subl = [];
prob.prisen.cons.subu = [1:7];
% analyse lower bound on variables 1:7
prob.prisen.vars.subl = [1:7];
prob.prisen.vars.subu = [];
% analyse 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 debug(100) 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 debug(100) echo(0)',prob,param);
results(2) = res;
%Print results
for m = [1:2]
  if m == 1
    fprintf('\nBasis sensitivity results:\n')
  else
    fprintf('\nOptimal partition sensitivity results:\n')
  end
  fprintf('\nSensitivity for bounds on constraints:\n')
  for i = 1:length(prob.prisen.cons.subl)
    fprintf (...
    'con = %d, beta_1 = %.1f, beta_2 = %.1f, delta_1 = %.1f,delta_2 = %.1f\n', ...
    prob.prisen.cons.subu(i),results(m).prisen.cons.lr_bu(i), ...
    results(m).prisen.cons.rr_bu(i),...
    results(m).prisen.cons.ls_bu(i),...
    results(m).prisen.cons.rs_bu(i));
  end

  for i = 1:length(prob.prisen.cons.subu)
    fprintf (...
    'con = %d, beta_1 = %.1f, beta_2 = %.1f, delta_1 = %.1f,delta_2 = %.1f\n', ...
    prob.prisen.cons.subu(i),results(m).prisen.cons.lr_bu(i), ...
    results(m).prisen.cons.rr_bu(i),...
    results(m).prisen.cons.ls_bu(i),...
    results(m).prisen.cons.rs_bu(i));
  end
  fprintf('Sensitivity for bounds on variables:\n')
  for i = 1:length(prob.prisen.vars.subl)
  fprintf (...
  'var = %d, beta_1 = %.1f, beta_2 = %.1f, delta_1 = %.1f,delta_2 = %.1f\n', ...
   prob.prisen.vars.subl(i),results(m).prisen.vars.lr_bl(i), ...
   results(m).prisen.vars.rr_bl(i),...
   results(m).prisen.vars.ls_bl(i),...
   results(m).prisen.vars.rs_bl(i));
  end

  for i = 1:length(prob.prisen.vars.subu)
    fprintf (...
    'var = %d, beta_1 = %.1f, beta_2 = %.1f, delta_1 = %.1f,delta_2 = %.1f\n', ...
    prob.prisen.vars.subu(i),results(m).prisen.vars.lr_bu(i), ...
    results(m).prisen.vars.rr_bu(i),...
    results(m).prisen.vars.ls_bu(i),...
    results(m).prisen.vars.rs_bu(i));
  end

  fprintf('Sensitivity for coefficients in objective:\n')
  for i = 1:length(prob.duasen.sub)
    fprintf (...
    'var = %d, beta_1 = %.1f, beta_2 = %.1f, delta_1 = %.1f,delta_2 = %.1f\n', ...
    prob.duasen.sub(i),results(m).duasen.lr_c(i), ...
    results(m).duasen.rr_c(i),...
    results(m).duasen.ls_c(i),...
    results(m).duasen.rs_c(i));
  end
end
```

The output from running the example `sensitivity.m` is shown below.

```
Basis sensitivity results:

Sensitivity for bounds on constraints:
con = 1, beta_1 = -300.0, beta_2 = 0.0, delta_1 = 3.0,delta_2 = 3.0
con = 2, beta_1 = -700.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0
con = 3, beta_1 = -500.0, beta_2 = 0.0, delta_1 = 3.0,delta_2 = 3.0
con = 4, beta_1 = -0.0, beta_2 = 500.0, delta_1 = 4.0,delta_2 = 4.0
con = 5, beta_1 = -0.0, beta_2 = 300.0, delta_1 = 5.0,delta_2 = 5.0
con = 6, beta_1 = -0.0, beta_2 = 700.0, delta_1 = 5.0,delta_2 = 5.0
con = 7, beta_1 = -500.0, beta_2 = 700.0, delta_1 = 2.0,delta_2 = 2.0
Sensitivity for bounds on variables:
var = 1, beta_1 = Inf, beta_2 = 300.0, delta_1 = 0.0,delta_2 = 0.0
var = 2, beta_1 = Inf, beta_2 = 100.0, delta_1 = 0.0,delta_2 = 0.0
var = 3, beta_1 = Inf, beta_2 = 0.0, delta_1 = 0.0,delta_2 = 0.0
var = 4, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 5, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 6, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 7, beta_1 = -0.0, beta_2 = 500.0, delta_1 = 2.0,delta_2 = 2.0
Sensitivity for coefficients in objective:
var = 1, beta_1 = Inf, beta_2 = 3.0, delta_1 = 300.0,delta_2 = 300.0
var = 2, beta_1 = Inf, beta_2 = Inf, delta_1 = 100.0,delta_2 = 100.0
var = 3, beta_1 = -2.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0
var = 4, beta_1 = Inf, beta_2 = 2.0, delta_1 = 500.0,delta_2 = 500.0
var = 5, beta_1 = -3.0, beta_2 = Inf, delta_1 = 500.0,delta_2 = 500.0
var = 6, beta_1 = Inf, beta_2 = 2.0, delta_1 = 500.0,delta_2 = 500.0
var = 7, beta_1 = -2.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0

Optimal partition sensitivity results:

Sensitivity for bounds on constraints:
con = 1, beta_1 = -300.0, beta_2 = 500.0, delta_1 = 3.0,delta_2 = 1.0
con = 2, beta_1 = -700.0, beta_2 = Inf, delta_1 = -0.0,delta_2 = -0.0
con = 3, beta_1 = -500.0, beta_2 = 500.0, delta_1 = 3.0,delta_2 = 1.0
con = 4, beta_1 = -500.0, beta_2 = 500.0, delta_1 = 2.0,delta_2 = 4.0
con = 5, beta_1 = -100.0, beta_2 = 300.0, delta_1 = 3.0,delta_2 = 5.0
con = 6, beta_1 = -500.0, beta_2 = 700.0, delta_1 = 3.0,delta_2 = 5.0
con = 7, beta_1 = -500.0, beta_2 = 700.0, delta_1 = 2.0,delta_2 = 2.0
Sensitivity for bounds on variables:
var = 1, beta_1 = Inf, beta_2 = 300.0, delta_1 = 0.0,delta_2 = 0.0
var = 2, beta_1 = Inf, beta_2 = 100.0, delta_1 = 0.0,delta_2 = 0.0
var = 3, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 2.0
var = 4, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 5, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 6, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 0.0
var = 7, beta_1 = Inf, beta_2 = 500.0, delta_1 = 0.0,delta_2 = 2.0
Sensitivity for coefficients in objective:
var = 1, beta_1 = Inf, beta_2 = 3.0, delta_1 = 300.0,delta_2 = 300.0
var = 2, beta_1 = Inf, beta_2 = Inf, delta_1 = 100.0,delta_2 = 100.0
var = 3, beta_1 = -2.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0
var = 4, beta_1 = Inf, beta_2 = 2.0, delta_1 = 500.0,delta_2 = 500.0
var = 5, beta_1 = -3.0, beta_2 = Inf, delta_1 = 500.0,delta_2 = 500.0
var = 6, beta_1 = Inf, beta_2 = 2.0, delta_1 = 500.0,delta_2 = 500.0
var = 7, beta_1 = -2.0, beta_2 = Inf, delta_1 = 0.0,delta_2 = 0.0
```

Appendix A

The MPS file format

MOSEK supports the standard MPS format with some extensions. For a detailed description of the MPS format the book by Nazareth [22] is a good reference.

A.1 The MPS file format

The version of the MPS format supported by MOSEK allows specification of an optimization problem on the form

$$\begin{array}{ccccc} l^c & \leq & Ax + q(x) & \leq & u^c, \\ l^x & \leq & x & \leq & u^x, \\ & & x \in \mathcal{C}, & & \\ & & x_{\mathcal{J}} \text{ integer}, & & \end{array} \tag{A.1}$$

where

- $x \in R^n$ is the vector of decision variables.
- $A \in R^{m \times n}$ is the constraint matrix.
- $l^c \in R^m$ is the lower limit on the activity for the constraints.
- $u^c \in R^m$ is the upper limit on the activity for the constraints.
- $l^x \in R^n$ is the lower limit on the activity for the variables.
- $u^x \in R^n$ is the upper limit on the activity for the variables.
- $q : R^n \rightarrow R$ is a vector of quadratic functions. Hence,

  $$q_i(x) = 1/2 x^T Q^i x$$

  where it is assumed that

  $$Q^i = (Q^i)^T. \tag{A.2}$$

Please note the explicit 1/2 in the quadratic term and that Q^i is required to be symmetric.

- $\mathcal{C}$ is a convex cone.
- $\mathcal{J} \subseteq \{1, 2, \ldots, n\}$ 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
*23456789012345678901234567890123456789012345678901234567890
NAME          [name]
OBJSENSE
    [objsense]
OBJNAME
    [objname]
ROWS
 ?  [cname1]
COLUMNS
    [vname1]  [cname1]    [value1]     [vname3]  [value2]
RHS
    [name]    [cname1]    [value1]     [cname2]  [value2]
RANGES
    [name]    [cname1]    [value1]     [cname2]  [value2]
QSECTION      [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 A.5 for details.

A.1.1 An example

A concrete example of a MPS file is presented below:

```
NAME          EXAMPLE
OBJSENSE
    MIN
ROWS
 N  obj
 L  c1
 L  c2
 L  c3
 L  c4
COLUMNS
    x1        obj       -10.0          c1        0.7
    x1        c2        0.5            c3        1.0
    x1        c4        0.1
    x2        obj       -9.0           c1        1.0
    x2        c2        0.8333333333   c3        0.66666667
    x2        c4        0.25
RHS
    rhs       c1        630.0          c2        600.0
    rhs       c3        708.0          c4        135.0
ENDATA
```

Subsequently each individual section in the MPS format is discussed.

A.1.2 `NAME`

In this section a name (`[name]`) is assigned to the problem.

A.1.3 `OBJSENSE` (optional)

This is an optional section that can be used to specify the sense of the objective function. The `OBJSENSE` section contains one line at most which can be one of the following

```
MIN
MINIMIZE
MAX
MAXIMIZE
```

It should be obvious what the implication is of each of these four lines.

A.1.4 `OBJNAME` (optional)

This is an optional section that can be used to specify the name of the row that is used as objective function. The `OBJNAME` section contains one line at most which has the form

```
objname
```

`objname` should be a valid row name.

A.1.5 `ROWS`

A record in the `ROWS` section has the form

```
?  [cname1]
```

where the requirements for the fields are as follows:

Field	Starting position	Maximum width	Required	Description
?	2	1	Yes	Constraint key
`[cname1]`	5	8	Yes	Constraint name

Hence, in this section each constraint is assigned an unique name denoted by `[cname1]`. Please note that `[cname1]` starts in position 5 and the field can be at most 8 characters wide. An initial key (?) must be present to specify the type of the constraint. The key can have the values `E`, `G`, `L`, or `N` whith ther following interpretation:

Constraint type	l_i^c	u_i^c
`E`	finite	l_i^c
`G`	finite	∞
`L`	$-\infty$	finite
`N`	$-\infty$	∞

In the MPS format an objective vector is not specified explicitly, but one of the constraints having the key `N` will be used as the objective vector c. In general, if multiple `N` type constraints are specified, then the first will be used as the objective vector c.

A.1.6 `COLUMNS`

In this section the elements of A are specified using one or more records having the form

```
  [vname1]  [cname1]    [value1]     [cname2]  [value2]
```

where the requirements for each field are as follows:

Field	Starting position	Maximum width	Required	Description
`[vname1]`	5	8	Yes	Variable name
`[cname1]`	15	8	Yes	Constraint name
`[value1]`	25	12	Yes	Numerical value
`[cname2]`	40	8	No	Constraint name
`[value2]`	50	12	No	Numerical value

Hence, a record specifies one or two elements a_{ij} of A using the principle that `[vname1]` and `[cname1]` determines j and i respectively. Please note that `[cname1]` must be a constraint name specified in the `ROWS` section. Finally, `[value1]` denotes the numerical value of a_{ij}. Another optional element is specified by `[cname2]`, and `[value2]` for the variable specified by `[vname1]`. Some important comments are:

- All elements belonging to one variable must be grouped together.
- Zero elements of A should not be specified.
- At least one element for each variable should be specified.

A.1.7 `RHS` (optional)

A record in this section has the format

```
  [name]    [cname1]    [value1]     [cname2]  [value2]
```

where the requirements for each field are as follows:

Field	Starting position	Maximum width	Re-quired	Description
`[name]`	5	8	Yes	Name of the `RHS` vector
`[cname1]`	15	8	Yes	Constraint name
`[value1]`	25	12	Yes	Numerical value
`[cname2]`	40	8	No	Constraint name
`[value2]`	50	12	No	Numerical value

The interpretation of a record is that `[name]` is the name of the `RHS` vector to be specified. In general, several vectors can be specified. `[cname1]` denotes a constraint name previously specified in the `ROWS` section. Now, assume that this name has been assigned to the ith constraint and v_1 denotes the value specified by `[value1]`, then the interpretation of v_1 is:

Constraint type	l_i^c	u_i^c
`E`	v_1	v_1
`G`	v_1	
`L`		v_1
`N`		

An optional second element is specified by `[cname2]` and `[value2]` and is interpreted in the same way. Please note that it is not necessary to specify zero elements, because elements are assumed to be zero.

A.1.8 `RANGES` (optional)

A record in this section has the form

```
    [name]     [cname1]     [value1]      [cname2]   [value2]
```

where the requirements for each fields are as follows:

Field	Starting position	Maximum width	Re-quired	Description
`[name]`	5	8	Yes	Name of the `RANGE` vector
`[cname1]`	15	8	Yes	Constraint name
`[value1]`	25	12	Yes	Numerical value
`[cname2]`	40	8	No	Constraint name
`[value2]`	50	12	No	Numerical value

The records in this section are used to modify the bound vectors for the constraints, i.e. the values in l^c and u^c. A record has the following interpretation: `[name]` is the name of the

`RANGE` vector anhd `[cname1]` is a valid constraint name. Assume that `[cname1]` is assigned to the ith 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 +		$l_i^c + \lvert v_1 \rvert$
`L`	- or +	$u_i^c - \lvert v_1 \rvert$	
`N`			

A.1.9 `QSECTION` (optional)

Within the `QSECTION` the label `[cname1]` must be a constraint name previously specified in the `ROWS` section. The label `[cname1]` denotes the constraint to which the quadratic term belongs. A record in the `QSECTION` has the form

```
[vname1]  [vname2]   [value1]    [vname3]  [value2]
```

where the requirements for each field are:

Field	Starting position	Maximum width	Required	Description
`[vname1]`	5	8	Yes	Variable name
`[vname2]`	15	8	Yes	Variable name
`[value1]`	25	12	Yes	Numerical value
`[vname3]`	40	8	No	Variable name
`[value2]`	50	12	No	Numerical value

A record specifies one or two elements in the lower triangular part of the Q^i matrix where `[cname1]` specifies the i. Hence, if the names `[vname1]` and `[vname2]` have been assigned to the kth and jth variable, then Q^i_{kj} is assigned the value given by `[value1]` An optional second element is specified in the same way by the fields `[vname1]`, `[vname3]`, and `[value2]`.

The example

$$
\begin{array}{lccc}
\text{minimize} & -x_2 + 0.5(2x_1^2 - 2x_1x_3 + 0.2x_2^2 + 2x_3^2) & & \\
\text{subject to} & x_1 + x_2 + x_3 & \geq & 1, \\
& x \geq 0 & &
\end{array}
$$

has the following MPS file representation

```
NAME          qoexp
ROWS
 N  obj
 G  c1
COLUMNS
    x1        c1        1
    x2        obj       -1
    x2        c1        1
    x3        c1        1
RHS
    rhs       c1        1
QSECTION      obj
    x1        x1        2
    x1        x3        -1
    x2        x2        0.2
    x3        x3        2
ENDATA
```

Regarding the `QSECTION`s please note that:

- Only one `QSECTION` is allowed for each constraint.
- The `QSECTION`s can appear in an arbitrary order after the `COLUMNS` section.
- All variable names occurring in the `QSECTION` must already be specified in the `COLUMNS` section.
- All entries specified in a `QSECTION` are assumed to belong to the lower triangular part of the quadratic term of Q.

A.1.10 `BOUNDS` (optional)

In the `BOUNDS` section changes to the default bounds vectors l^x and u^x are specified. The default bounds vectors are $l^x = 0$ and $u^x = \infty$. Moreover, it is possible to specify several sets of bound vectors. A record in this section has the form

```
?? [name]    [vname1]    [value1]
```

where the requirements for each field are:

Field	Starting position	Maximum width	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	Variable name

Hence, a record in the `BOUNDS` section has the following interpretation: `[name]` is the name of the bound vector and `[vname1]` is the name of the variable which bounds are modified by the record. `??` and `[value1]` are used to modify the bound vectors according to the following table:

`??`	l_j^x	u_j^x	Made integer (added to $\mathcal{J}$)
`FR`	$-\infty$	∞	No
`FX`	v_1	v_1	No
`LO`	v_1	unchanged	No
`MI`	$-\infty$	unchanged	No
`PL`	unchanged	∞	No
`UP`	unchanged	v_1	No
`BV`	0	1	Yes
`LI`	$\lceil v_1 \rceil$	∞	Yes
`UI`	unchanged	$\lfloor v_1 \rfloor$	Yes

v_1 is the value specified by `[value1]`.

A.1.11 `CSECTION` (optional)

The purpose of the `CSECTION` is to specify the constraint

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

in (A.1).

It is assumed that $\mathcal{C}$ satisfies the following requirements. Let

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

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

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

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

- R set:
$$\mathcal{C}_t = \{x \in R^{n^t}\}.$$
- Quadratic cone:
$$\mathcal{C}_t = \left\{ x \in R^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\}. \tag{A.3}$$
- Rotated quadratic cone:
$$\mathcal{C}_t = \left\{ x \in R^{n^t} : 2x_1x_2 \geq \sum_{j=3}^{n^t} x_j^2, \ x_1, x_2 \geq 0 \right\}. \tag{A.4}$$

In general, only quadratic and rotated quadratic cones are specified in the MPS file whereas membership of the 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 R cone.

Next, let us study an example. Assume that the quadratic cone

$$x_4 \geq \sqrt{x_5^2 + x_8^2} \tag{A.5}$$

and the rotated quadratic cone

$$2x_3x_7 \geq x_1^2 + x_8^2, \ x_3, x_7 \geq 0, \tag{A.6}$$

should be specified in the MPS file. One `CSECTION` is required for each cone and they are specified as follows:

```
*        1         2         3         4         5         6
*234567890123456789012345678901234567890123456789012345678901234567890
CSECTION      konea       0.0          QUAD
    x4
    x5
    x8
CSECTION      koneb       0.0          RQUAD
    x7
    x3
    x1
    x0
```

This first `CSECTION` specifies the cone (A.5) which is given the name `konea`. This is a quadratic cone which is specified by the keyword `QUAD` in the `CSECTION` header. The 0.0 value in the `CSECTION` header is not used by the `QUAD` cone.

The second `CSECTION` specifies the rotated quadratic cone (A.6). Please note the keyword `RQUAD` in the `CSECTION` which is used to specify that the cone is a rotated quadratic cone instead of a quadratic cone. The 0.0 value in the `CSECTION` header is not used by the `RQUAD` cone.

In general, a `CSECTION` header has the format

```
CSECTION      [kname1]    [value1]     [ktype]
```

where the requirement for each field are as follows:

Field	Starting position	Maximum width	Required	Description
`[kname1]`	5	8	Yes	Name of the cone
`[value1]`	15	12	No	Cone parameter
`[ktype]`	25		Yes	Type of the cone.

The possible cone type keys are:

Cone type key	Members	Interpretation.
`QUAD`	≥ 1	Quadratic cone i.e. (A.3).
`RQUAD`	≥ 2	Rotated quadratic cone i.e. (A.4).

Please note that a quadratic cone must have at least one member whereas a rotated quadratic cone must have at least two members. A record in the `CSECTION` has the format

```
 [vname1]
```

where the requirements for each field are

Field	Starting position	Maximum width	Required	Description
`[vname1]`	2	8	Yes	A valid variable name

The most important restriction with respect to the `CSECTION` is that a variable must occur in only one `CSECTION`.

A.1.12 `ENDATA`

This keyword denotes the end of the MPS file.

A.2 Integer variables

Using special bound keys in the `BOUNDS` section it is possible to specify that some or all of the variables should be integer constrained i.e. be members of $\mathcal{J}$. However, an alternative method is available.

This method is available only for backward compability and we recommend that it is not used. This method requires that markers are placed in the `COLUMNS` section as in the example:

```
COLUMNS
    x1         obj        -10.0          c1         0.7
    x1         c2         0.5            c3         1.0
    x1         c4         0.1
* Start of integer constrained variables.
    MARK000    'MARKER'                  'INTORG'
    x2         obj        -9.0           c1         1.0
    x2         c2         0.8333333333   c3         0.66666667
    x2         c4         0.25
    x3         obj        1.0            c6         2.0
    MARK001    'MARKER'                  'INTEND'
* End of integer constrained variables.
```

Please note that special marker lines are used to indicate the start and the end of the integer variables. Furthermore be aware of the following

- IMPORTANT: All variables between the markers are assigned a default lower bound of 0 and a default upper bound of 1. **This may not be what is intended.** If it is not intended, the correct bounds should be defined in the `BOUNDS` section of the MPS formatted file.
- MOSEK ignores field 1, i.e. `MARK0001` and `MARK001`, however, other optimization systems require them.
- Field 2, i.e. `'MARKER'`, must be specified including the single quotes. This implies that no row can be assigned the name `'MARKER'`.
- Field 3 is ignored and should be left blank.
- Field 4, i.e. `'INTORG'` and `'INTEND'`, must be specified.
- It is possible to specify several such integer marker sections within the `COLUMNS` section.

A.3 General limitations

- An MPS file should be an ASCII file.

A.4 Interpretation of the MPS format

Several issues related to the MPS format are not well-defined by the industry standard. However, MOSEK uses the following interpretation:

- If a matrix element in the `COLUMNS` section is specified multiple times, then the multiple entries are added together.
- If a matrix element in a `QSECTION` section is specified multiple times, then the multiple entries are added together.

A.5 The free MPS format

MOSEK supports a free format variation of the MPS format. The free format is similar to the MPS file format but less restrictive, e.g. it allows longer names. However, it also presents two main limitations:

- By default a line in the MPS file must not contain more than 1024 characters. However, by modifying the parameter `MSK_IPAR_READ_MPS_WIDTH` an arbitrary large line width will be accepted.
- A name must not contain any blanks.

To use the free MPS format instead of the default MPS format the MOSEK parameter `MSK_IPAR_READ_MPS_FORMAT` should be changed.

Appendix B

The LP file format

MOSEK supports the LP file format with some extensions i.e. MOSEK can read and write LP formatted files.

B.1 A warning

The LP format is not a well-defined standard and hence different optimization packages may interpretate a specific LP formatted file differently.

B.2 The LP file format

The LP file format can specify problems on the form

$$
\begin{array}{lccccc}
\text{minimize/maximize} & & & c^T x + \frac{1}{2} q^o(x) & & \\
\text{subject to} & l^c & \leq & Ax + \frac{1}{2} q(x) & \leq & u^c, \\
 & l^x & \leq & x & \leq & u^x, \\
 & & & x_{\mathcal{J}} \text{ integer}, & &
\end{array}
$$

where

- $x \in R^n$ is the vector of decision variables.
- $c \in R^n$ is the linear term in the objective.
- $q^o :\in R^n \rightarrow R$ is the quadratic term in the objective where

  $$q^o(x) = x^T Q^o x$$

  and it is assumed that

  $$Q^o = (Q^o)^T. \tag{B.1}$$

- $A \in R^{m \times n}$ is the constraint matrix.
- $l^c \in R^m$ is the lower limit on the activity for the constraints.
- $u^c \in R^m$ is the upper limit on the activity for the constraints.
- $l^x \in R^n$ is the lower limit on the activity for the variables.
- $u^x \in R^n$ is the upper limit on the activity for the variables.
- $q : R^n \rightarrow R$ is a vector of quadratic functions. Hence,

  $$q_i(x) = x^T Q^i x$$

  where it is assumed that

  $$Q^i = (Q^i)^T. \tag{B.2}$$

- $\mathcal{J} \subseteq \{1, 2, \ldots, n\}$ is an index set of the integer constrained variables.

B.2.1 The sections

An LP formatted file contains a number of sections specifying the objective, constraints, variable bounds, and variable types. The section keywords may be any mix of upper and lower case letters.

B.2.1.1 The objective

The first section beginning with one of the keywords

```
max
maximum
maximize
min
minimum
minimize
```

defines the objective sense and the objective function, i.e.

$$c^T x + \frac{1}{2} x^T Q^o x.$$

The objective may be given a name by writing

```
myname:
```

before the expressions. If no name is given, then the objective is named `obj`.

The objective function contains linear and quadratic terms. The linear terms are written as in the example

```
4 x1 + x2 - 0.1 x3
```

and so forth. The quadratic terms are written in square brackets (`[ ]`) and are either squared or multiplied as in the examples

```
x1 ^ 2
```

and

```
x1 * x2
```

There may be zero or more pairs of brackets containing quadratic expressions.

An example of an objective section is:

```
minimize
  myobj: 4 x1 + x2 - 0.1 x3 + [ x1 ^ 2 + 2.1 x1 * x2 ]/2
```

Please note that the quadratic expressions are multiplied with $\frac{1}{2}$, so that the above expression means

$$\text{minimize } 4x_1 + x_2 - 0.1 \cdot x_3 + \frac{1}{2}(x_1^2 + 2.1 \cdot x_1 \cdot x_2)$$

If the same variable occurs more than once in the linear part, the coefficients are added, so that `4 x1 + 2 x1` is equivalent to `6 x1`. In the quadratic expressions `x1 * x2` is equivalent to `x2 * x1` and as in the linear part , if the same variables multiplied or squared occur several times their coefficients are added.

B.2.1.2 The constraints

The second section beginning with one of the keywords

```
subj to
subject to
s.t.
st
```

defines the linear constraint matrix (A) and the quadratic matrices (Q^i).

A constraint contains a name (optional), expressions adhering to the same rules as in the objective and a bound:

```
subject to
  con1: x1 + x2 + [ x3 ^ 2 ]/2 <= 5.1
```

The bound type (here `<=`) 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.

$$-5 \leq x_1 + x_2 \leq 5 \tag{B.3}$$

may be written as

```
con:: -5 < x_1 + x_2 < 5
```

By default MOSEK writes ranged constraints this way.

If the files must adhere to the LP standard, ranged constraints must either be split into upper bounded and lower bounded constraints or be written as en equality with a slack variable. For example the expression (B.3) may be written as

$$x_1 + x_2 - sl_1 = 0, \ -5 \leq sl_1 \leq 5.$$

B.2.1.3 Bounds

Bounds on the variables can be specified in the bound section beginning with one of the keywords

```
bound
bounds
```

The bounds section is optional but should, if present, follow the `subject to` section. All variables listed in the bounds section must occur in either the objective or a constraint.

The default lower and upper bounds are 0 and $+\infty$. A variable may be declared free with the keyword `free`, which means that the lower bound is $-\infty$ and the upper bound is $+\infty$. Furthermore it may be assigned a finite lower and upper bound. The bound definitions for a given variable may be written in one or two lines, and bounds can be any number or $\pm\infty$ (written as `+inf`/`-inf`/`+infinity`/`-infinity`) as in the example

```
bounds
  x1 free
  x2 <= 5
  0.1 <= x2
  x3 = 42
  2 <= x4 < +inf
```

B.2.1.4 Variable types

The final two sections are optional and must begin with one of the keywords

```
bin
binaries
binary
```

and

```
gen
general
```

Under `general` all integer variables are listed, and under `binary` all binary (integer variables with bounds 0 and 1) are listed:

```
general
  x1 x2
binary
  x3 x4
```

Again, all variables listed in the binary or general sections must occur in either the objective or a constraint.

B.2.1.5 Terminating section

Finally, an LP formatted file must be terminated with the keyword

```
end
```

B.2.1.6 An example

A simple example of an LP file with two variables, four constraints and one integer variable is:

```
minimize
  -10 x1 -9 x2
subject to
  0.7 x1 +       x2 <= 630
  0.5 x1 + 0.833 x2 <= 600
      x1 + 0.667 x2 <= 708
  0.1 x1 + 0.025 x2 <= 135
bounds
  10 <= x1
  x1 <= +inf
  20 <= x2 <= 500
general
  x1
end
```

B.2.2 LP format peculiarities

B.2.2.1 Comments

Anything on a line after a “\” is ignored and is treated as a comment.

B.2.2.2 Names

A name for an objective, a constraint or a variable may contain the letters a-z, A-Z, the digits 0-9 and the characters

```
!"#$%&()/,.;?@_’‘{}|~
```

The first character in a name must not be a number, a period or the letter ’`e`’ or ’`E`’. Keywords must not be used as names.

It is strongly recommended not to use double quotes (`"`) in names.

B.2.2.3 Variable bounds

Specifying several upper or lower bounds on one variable is possible but MOSEK uses only the tightest bounds. If a variable is fixed (with `=`), then it is considered the tightest bound.

B.2.2.4 MOSEK specific extensions to the LP format

Some optimization software packages employ a more strict definition of the LP format 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.

B.2.3 The strict LP format

The LP format is not a formal standard and different vendors have slightly different interpretations of the LP format. To make MOSEK's definition of the LP format more compatible whith the definitions of other vendors use the paramter 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 paramter setting

```
MSK_IPAR_WRITE_GENERIC_NAMES MSK_ON
```

which will cause all names to be renamed systematically in the output file.

B.2.4 Formatting of an LP file

A few parameters control the visual formatting of LP files written by MOSEK in order to make it easier to read the files. These parameters are

```
MSK_IPAR_WRITE_LP_LINE_WIDTH
MSK_IPAR_WRITE_LP_TERMS_PER_LINE
```

The first parameter sets the maximum number of characters on a single line. The default value is 80 corresponding roughly to the width of a standard text document.

The second parameter sets the maximum number of terms per line; a term means a sign, a coefficient, and a name (for example "`+ 42 elephants`"). The default value is 0, meaning that there is no maximum.

B.2.4.1 Speeding up file reading

If the input file should be read as fast as possible using the least amount of memory, then it is important to tell MOSEK how many non-zeros, variables and constraints the problem contains. These values can be set using the parameters

```
MSK_IPAR_READ_CON
MSK_IPAR_READ_VAR
MSK_IPAR_READ_ANZ
MSK_IPAR_READ_QNZ
```

B.2.4.2 Unnamed constraints

Reading and writing an LP file with MOSEK may change it superficially. If an LP file contains unnamed constraints or objective these are given their generic names when the file is read (however unnamed constraints in MOSEK are written without names).

Appendix C

Parameters

Subsequently all parameters that are in MOSEK parameter database is presented. For each parameter their name, purpose, type, default value etc. are presented.

C.1 Parameter groups

Parameters grouped by meaning and functionality.

C.1.1 Logging parameters.

- `MSK_IPAR_LOG` 282
  Controls the amount of log information.

- `MSK_IPAR_LOG_BI` 282
  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` 282
  Controls the logging frequency.

- `MSK_IPAR_LOG_CONCURRENT` 283
  Controls amount of output printed by the concurrent optimizer.

- `MSK_IPAR_LOG_CUT_SECOND_OPT` 283
  Controls the reduction in the log levels for the second and any subsequent optimizations.

- `MSK_IPAR_LOG_FACTOR` 283
  If turned on, then the factor log lines are added to the log.

- `MSK_IPAR_LOG_FEASREPAIR` 284
  Controls the amount of output printed when performing feasibility repair.

- MSK_IPAR_LOG_FILE 284
  If turned on, then some log info is printed when a file is written or read.

- MSK_IPAR_LOG_HEAD 284
  If turned on, then a header line is added to the log.

- MSK_IPAR_LOG_INFEAS_ANA 285
  Controls log level for the infeasibility analyzer.

- MSK_IPAR_LOG_INTPNT 285
  Controls the amount of log information from the interior-point optimizers.

- MSK_IPAR_LOG_MIO 285
  Controls the amount of log information from the mixed-integer optimizers.

- MSK_IPAR_LOG_MIO_FREQ 285
  The mixed integer solver logging frequency.

- MSK_IPAR_LOG_NONCONVEX 286
  Controls amount of output printed by the nonconvex optimizer.

- MSK_IPAR_LOG_OPTIMIZER 286
  Controls the amount of general optimizer information that is logged.

- MSK_IPAR_LOG_ORDER 286
  If turned on, then factor lines are added to the log.

- MSK_IPAR_LOG_PARAM 287
  Controls the amount of information printed out about parameter changes.

- MSK_IPAR_LOG_PRESOLVE 287
  Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.

- MSK_IPAR_LOG_RESPONSE 287
  Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.

- MSK_IPAR_LOG_SENSITIVITY 287
  Control logging in sensitivity analyzer.

- MSK_IPAR_LOG_SENSITIVITY_OPT 288
  Control logging in sensitivity analyzer.

- MSK_IPAR_LOG_SIM 288
  Controls the amount of log information from the simplex optimizers.

- `MSK_IPAR_LOG_SIM_FREQ` 288
  Controls simplex logging frequency.

- `MSK_IPAR_LOG_SIM_NETWORK_FREQ` 289
  Controls the network simplex logging frequency.

- `MSK_IPAR_LOG_STORAGE` 289
  Controls the memory related log information.

C.1.2 Basis identification parameters.

- `MSK_IPAR_BI_CLEAN_OPTIMIZER` 267
  Controls which simplex optimizer is used in the clean-up phase.

- `MSK_IPAR_BI_IGNORE_MAX_ITER` 267
  Turns on basis identification in case the interior-point optimizer is terminated due to maximum number of iterations.

- `MSK_IPAR_BI_IGNORE_NUM_ERROR` 268
  Turns on basis identification in case the interior-point optimizer is terminated due to a numerical problem.

- `MSK_DPAR_BI_LU_TOL_REL_PIV` 236
  Relative pivot tolerance used in the LU factorization in the basis identification procedure.

- `MSK_IPAR_BI_MAX_ITERATIONS` 268
  Maximum number of iterations after basis identification.

- `MSK_IPAR_INTPNT_BASIS` 274
  Controls whether basis identification is performed.

- `MSK_IPAR_LOG_BI` 282
  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` 282
  Controls the logging frequency.

C.1.3 The Interior-point method parameters.

Parameters defining the behavior of the interior-point method for linear, conic and convex problems.

- `MSK_IPAR_BI_IGNORE_MAX_ITER` 267
  Turns on basis identification in case the interior-point optimizer is terminated due to maximum number of iterations.

- `MSK_IPAR_BI_IGNORE_NUM_ERROR` 268
  Turns on basis identification in case the interior-point optimizer is terminated due to a numerical problem.

- `MSK_IPAR_INTPNT_BASIS` 274
  Controls whether basis identification is performed.

- `MSK_DPAR_INTPNT_CO_TOL_DFEAS` 240
  Dual feasibility tolerance used by the conic interior-point optimizer.

- `MSK_DPAR_INTPNT_CO_TOL_INFEAS` 240
  Infeasibility tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_MU_RED` 240
  Optimality tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_NEAR_REL` 241
  Optimality tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_PFEAS` 241
  Primal feasibility tolerance used by the conic interior-point optimizer.

- `MSK_DPAR_INTPNT_CO_TOL_REL_GAP` 241
  Relative gap termination tolerance used by the conic interior-point optimizer.

- `MSK_IPAR_INTPNT_DIFF_STEP` 275
  Controls whether different step sizes are allowed in the primal and dual space.

- `MSK_IPAR_INTPNT_MAX_ITERATIONS` 276
  Controls the maximum number of iterations allowed in the interior-point optimizer.

- `MSK_IPAR_INTPNT_MAX_NUM_COR` 276
  Maximum number of correction steps.

- `MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS` 277
  Maximum number of steps to be used by the iterative search direction refinement.

- `MSK_DPAR_INTPNT_NL_MERIT_BAL` 242
  Controls if the complementarity and infeasibility is converging to zero at about equal rates.

- MSK_DPAR_INTPNT_NL_TOL_DFEAS 242
  Dual feasibility tolerance used when a nonlinear model is solved.
- MSK_DPAR_INTPNT_NL_TOL_MU_RED 242
  Relative complementarity gap tolerance.
- MSK_DPAR_INTPNT_NL_TOL_NEAR_REL 242
  Nonlinear solver optimality tolerance parameter.
- MSK_DPAR_INTPNT_NL_TOL_PFEAS 243
  Primal feasibility tolerance used when a nonlinear model is solved.
- MSK_DPAR_INTPNT_NL_TOL_REL_GAP 243
  Relative gap termination tolerance for nonlinear problems.
- MSK_DPAR_INTPNT_NL_TOL_REL_STEP 243
  Relative step size to the boundary for general nonlinear optimization problems.
- MSK_IPAR_INTPNT_OFF_COL_TRH 277
  Controls the aggressiveness of the offending column detection.
- MSK_IPAR_INTPNT_ORDER_METHOD 277
  Controls the ordering strategy.
- MSK_IPAR_INTPNT_REGULARIZATION_USE 278
  Controls whether regularization is allowed.
- MSK_IPAR_INTPNT_SCALING 278
  Controls how the problem is scaled before the interior-point optimizer is used.
- MSK_IPAR_INTPNT_SOLVE_FORM 279
  Controls whether the primal or the dual problem is solved.
- MSK_IPAR_INTPNT_STARTING_POINT 279
  Starting point used by the interior-point optimizer.
- MSK_DPAR_INTPNT_TOL_DFEAS 244
  Dual feasibility tolerance used for linear and quadratic optimization problems.
- MSK_DPAR_INTPNT_TOL_DSAFE 244
  Controls the interior-point dual starting point.
- MSK_DPAR_INTPNT_TOL_INFEAS 244
  Nonlinear solver infeasibility tolerance parameter.
- MSK_DPAR_INTPNT_TOL_MU_RED 244
  Relative complementarity gap tolerance.

- MSK_DPAR_INTPNT_TOL_PATH 245
  interior-point centering aggressiveness.

- MSK_DPAR_INTPNT_TOL_PFEAS 245
  Primal feasibility tolerance used for linear and quadratic optimization problems.

- MSK_DPAR_INTPNT_TOL_PSAFE 245
  Controls the interior-point primal starting point.

- MSK_DPAR_INTPNT_TOL_REL_GAP 246
  Relative gap termination tolerance.

- MSK_DPAR_INTPNT_TOL_REL_STEP 246
  Relative step size to the boundary for linear and quadratic optimization problems.

- MSK_DPAR_INTPNT_TOL_STEP_SIZE 246
  If the step size falls below the value of this parameter, then the interior-point optimizer assumes it is stalled. It it does not not make any progress.

- MSK_IPAR_LOG_CONCURRENT 283
  Controls amount of output printed by the concurrent optimizer.

- MSK_IPAR_LOG_INTPNT 285
  Controls the amount of log information from the interior-point optimizers.

- MSK_IPAR_LOG_PRESOLVE 287
  Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.

C.1.4 Simplex optimizer parameters.

Parameters defining the behavior of the simplex optimizer for linear problems.

- MSK_DPAR_BASIS_REL_TOL_S 236
  Maximum relative dual bound violation allowed in an optimal basic solution.

- MSK_DPAR_BASIS_TOL_S 236
  Maximum absolute dual bound violation in an optimal basic solution.

- MSK_DPAR_BASIS_TOL_X 236
  Maximum absolute primal bound violation allowed in an optimal basic solution.

- MSK_IPAR_LOG_SIM 288
  Controls the amount of log information from the simplex optimizers.

- MSK_IPAR_LOG_SIM_FREQ 288
  Controls simplex logging frequency.
- MSK_IPAR_LOG_SIM_MINOR 289
  Currently not in use.
- MSK_IPAR_SENSITIVITY_OPTIMIZER 313
  Controls which optimizer is used for optimal partition sensitivity analysis.
- MSK_IPAR_SIM_DEGEN 314
  Controls how aggressive degeneration is approached.
- MSK_IPAR_SIM_HOTSTART 315
  Controls the type of hot-start that the simplex optimizer perform.
- MSK_IPAR_SIM_MAX_ITERATIONS 316
  Maximum number of iterations that can be used by a simplex optimizer.
- MSK_IPAR_SIM_MAX_NUM_SETBACKS 316
  Controls how many setbacks that are allowed within a simplex optimizer.
- MSK_IPAR_SIM_NETWORK_DETECT_METHOD 317
  Controls which type of detection method the network extraction should use.
- MSK_IPAR_SIM_NON_SINGULAR 317
  Controls if the simplex optimizer ensures a non-singular basis, if possible.
- MSK_IPAR_SIM_SAVE_LU 319
  Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.
- MSK_IPAR_SIM_SCALING 320
  Controls how the problem is scaled before a simplex optimizer is used.
- MSK_IPAR_SIM_SOLVE_FORM 320
  Controls whether the primal or the dual problem is solved by the primal-/dual- simplex optimizer.
- MSK_IPAR_SIM_STABILITY_PRIORITY 320
  Controls how high priority the numerical stability should be given.
- MSK_IPAR_SIM_SWITCH_OPTIMIZER 321
  Controls the simplex behavior.
- MSK_DPAR_SIMPLEX_ABS_TOL_PIV 253
  Absolute pivot tolerance employed by the simplex optimizers.

C.1.5 Primal simplex optimizer parameters.

Parameters defining the behavior of the primal simplex optimizer for linear problems.

- MSK_IPAR_SIM_PRIMAL_CRASH 318
  Controls the simplex crash.

- MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION 318
  Controls how aggressively restricted selection is used.

- MSK_IPAR_SIM_PRIMAL_SELECTION 318
  Controls the primal simplex strategy.

C.1.6 Dual simplex optimizer parameters.

Parameters defining the behavior of the dual simplex optimizer for linear problems.

- MSK_IPAR_SIM_DUAL_CRASH 314
  Controls whether crashing is performed in the dual simplex optimizer.

- MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION 314
  Controls how aggressively restricted selection is used.

- MSK_IPAR_SIM_DUAL_SELECTION 315
  Controls the dual simplex strategy.

C.1.7 Network simplex optimizer parameters.

Parameters defining the behavior of the network simplex optimizer for linear problems.

- MSK_IPAR_LOG_SIM_NETWORK_FREQ 289
  Controls the network simplex logging frequency.

- MSK_IPAR_SIM_NETWORK_DETECT 316
  Level of aggressiveness of network detection.

- MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART 317
  Level of aggressiveness of network detection in a simplex hot-start.

- MSK_IPAR_SIM_REFACTOR_FREQ 319
  Controls the basis refactoring frequency.

C.1.8 Nonlinear convex method parameters.

Parameters defining the behavior of the interior-point method for nonlinear convex problems.

- MSK_IPAR_CHECK_CONVEXITY 269
  Specify the level of convexity check on quadratic problems

- MSK_DPAR_INTPNT_NL_MERIT_BAL 242
  Controls if the complementarity and infeasibility is converging to zero at about equal rates.

- MSK_DPAR_INTPNT_NL_TOL_DFEAS 242
  Dual feasibility tolerance used when a nonlinear model is solved.

- MSK_DPAR_INTPNT_NL_TOL_MU_RED 242
  Relative complementarity gap tolerance.

- MSK_DPAR_INTPNT_NL_TOL_NEAR_REL 242
  Nonlinear solver optimality tolerance parameter.

- MSK_DPAR_INTPNT_NL_TOL_PFEAS 243
  Primal feasibility tolerance used when a nonlinear model is solved.

- MSK_DPAR_INTPNT_NL_TOL_REL_GAP 243
  Relative gap termination tolerance for nonlinear problems.

- MSK_DPAR_INTPNT_NL_TOL_REL_STEP 243
  Relative step size to the boundary for general nonlinear optimization problems.

- MSK_DPAR_INTPNT_TOL_INFEAS 244
  Nonlinear solver infeasibility tolerance parameter.

C.1.9 The conic interior-point method parameters.

Parameters defining the behavior of the interior-point method for conic problems.

- MSK_DPAR_INTPNT_CO_TOL_DFEAS 240
  Dual feasibility tolerance used by the conic interior-point optimizer.

- MSK_DPAR_INTPNT_CO_TOL_INFEAS 240
  Infeasibility tolerance for the conic solver.

- MSK_DPAR_INTPNT_CO_TOL_MU_RED 240
  Optimality tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_NEAR_REL` 241
  Optimality tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_PFEAS` 241
  Primal feasibility tolerance used by the conic interior-point optimizer.

- `MSK_DPAR_INTPNT_CO_TOL_REL_GAP` 241
  Relative gap termination tolerance used by the conic interior-point optimizer.

C.1.10 The mixed integer optimization parameters.

- `MSK_IPAR_LOG_MIO` 285
  Controls the amount of log information from the mixed-integer optimizers.

- `MSK_IPAR_LOG_MIO_FREQ` 285
  The mixed integer solver logging frequency.

- `MSK_IPAR_MIO_BRANCH_DIR` 290
  Controls whether the mixed integer optimizer is branching up or down by default.

- `MSK_IPAR_MIO_BRANCH_PRIORITIES_USE` 291
  Controls whether branching priorities are used by the mixed integer optimizer.

- `MSK_IPAR_MIO_CONSTRUCT_SOL` 291
  Controls if initial MIP solution should be constructed from value of integer variables.

- `MSK_IPAR_MIO_CONT_SOL` 291
  Controls the meaning of interior-point and basic solutions in MIP problems.

- `MSK_IPAR_MIO_CUT_LEVEL_ROOT` 292
  Controls the cut level employed by the mixed integer optimizer at the root node.

- `MSK_IPAR_MIO_CUT_LEVEL_TREE` 293
  Controls the cut level employed by the mixed integer optimizer in the tree.

- `MSK_DPAR_MIO_DISABLE_TERM_TIME` 247
  Certain termination criterias is disabled within the mixed integer optimizer for period time specified by the parameter.

- `MSK_IPAR_MIO_FEASPUMP_LEVEL` 293
  Controls the feasibility pump heuristic which is used to construct a good initial feasible solution.

- `MSK_IPAR_MIO_HEURISTIC_LEVEL` 293
  Controls the heuristic employed by the mixed integer optimizer to locate an initial integer feasible solution.

- MSK_DPAR_MIO_HEURISTIC_TIME 248
  Time limit for the mixed integer heuristics.

- MSK_IPAR_MIO_KEEP_BASIS 294
  Controls whether the integer presolve keeps bases in memory.

- MSK_IPAR_MIO_MAX_NUM_BRANCHES 294
  Maximum number of branches allowed during the branch and bound search.

- MSK_IPAR_MIO_MAX_NUM_RELAXS 295
  Maximum number of relaxations in branch and bound search.

- MSK_IPAR_MIO_MAX_NUM_SOLUTIONS 295
  Controls how many feasible solutions the mixed-integer optimizer investigates.

- MSK_DPAR_MIO_MAX_TIME 248
  Time limit for the mixed integer optimizer.

- MSK_DPAR_MIO_MAX_TIME_APRX_OPT 248
  Time limit for the mixed integer optimizer.

- MSK_DPAR_MIO_NEAR_TOL_ABS_GAP 249
  Relaxed absolute optimality tolerance employed by the mixed integer optimizer.

- MSK_DPAR_MIO_NEAR_TOL_REL_GAP 249
  The mixed integer optimizer is terminated when this tolerance is satisfied.

- MSK_IPAR_MIO_NODE_OPTIMIZER 296
  Controls which optimizer is employed at the non-root nodes in the mixed integer optimizer.

- MSK_IPAR_MIO_NODE_SELECTION 296
  Controls the node selection strategy employed by the mixed integer optimizer.

- MSK_IPAR_MIO_PRESOLVE_AGGREGATE 297
  Controls whether problem aggregation is performed in the mixed integer presolve.

- MSK_IPAR_MIO_PRESOLVE_PROBING 297
  Controls whether probing is employed by the mixed integer presolve.

- MSK_IPAR_MIO_PRESOLVE_USE 298
  Controls whether presolve is performed by the mixed integer optimizer.

- MSK_DPAR_MIO_REL_ADD_CUT_LIMITED 249
  Controls cut generation for MIP solver.

- `MSK_IPAR_MIO_ROOT_OPTIMIZER` ... 298
  Controls which optimizer is employed at the root node in the mixed integer optimizer.

- `MSK_IPAR_MIO_STRONG_BRANCH` ... 298
  The depth from the root in which strong branching is employed.

- `MSK_DPAR_MIO_TOL_ABS_GAP` ... 250
  Absolute optimality tolerance employed by the mixed integer optimizer.

- `MSK_DPAR_MIO_TOL_ABS_RELAX_INT` ... 250
  Integer constraint tolerance.

- `MSK_DPAR_MIO_TOL_REL_GAP` ... 250
  Relative optimality tolerance employed by the mixed integer optimizer.

- `MSK_DPAR_MIO_TOL_REL_RELAX_INT` ... 251
  Integer constraint tolerance.

- `MSK_DPAR_MIO_TOL_X` ... 251
  Absolute solution tolerance used in mixed-integer optimizer.

C.1.11 Presolve parameters.

- `MSK_IPAR_PRESOLVE_ELIM_FILL` ... 303
  Maximum amount of fill-in in the elimination phase.

- `MSK_IPAR_PRESOLVE_ELIMINATOR_USE` ... 304
  Controls whether free or implied free variables are eliminated from the problem.

- `MSK_IPAR_PRESOLVE_LEVEL` ... 304
  Currently not used.

- `MSK_IPAR_PRESOLVE_LINDEP_USE` ... 304
  Controls whether the linear constraints are checked for linear dependencies.

- `MSK_IPAR_PRESOLVE_LINDEP_WORK_LIM` ... 305
  Controls linear dependency check in presolve.

- `MSK_DPAR_PRESOLVE_TOL_AIJ` ... 252
  Absolute zero tolerance employed for constraint coefficients in the presolve.

- `MSK_DPAR_PRESOLVE_TOL_LIN_DEP` ... 252
  Controls when a constraint is determined to be linearly dependent.

- `MSK_DPAR_PRESOLVE_TOL_S` ... 253
  Absolute zero tolerance employed for slack variables in the presolve.

- `MSK_DPAR_PRESOLVE_TOL_X` 253
  Absolute zero tolerance employed for variables in the presolve.

- `MSK_IPAR_PRESOLVE_USE` 305
  Controls whether the presolve is applied to a problem before it is optimized.

C.1.12 Termination criterion parameters.

Parameters which define termination and optimality criteria and related information.

- `MSK_DPAR_BASIS_REL_TOL_S` 236
  Maximum relative dual bound violation allowed in an optimal basic solution.

- `MSK_DPAR_BASIS_TOL_S` 236
  Maximum absolute dual bound violation in an optimal basic solution.

- `MSK_DPAR_BASIS_TOL_X` 236
  Maximum absolute primal bound violation allowed in an optimal basic solution.

- `MSK_IPAR_BI_MAX_ITERATIONS` 268
  Maximum number of iterations after basis identification.

- `MSK_DPAR_INTPNT_CO_TOL_DFEAS` 240
  Dual feasibility tolerance used by the conic interior-point optimizer.

- `MSK_DPAR_INTPNT_CO_TOL_INFEAS` 240
  Infeasibility tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_MU_RED` 240
  Optimality tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_NEAR_REL` 241
  Optimality tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_PFEAS` 241
  Primal feasibility tolerance used by the conic interior-point optimizer.

- `MSK_DPAR_INTPNT_CO_TOL_REL_GAP` 241
  Relative gap termination tolerance used by the conic interior-point optimizer.

- `MSK_IPAR_INTPNT_MAX_ITERATIONS` 276
  Controls the maximum number of iterations allowed in the interior-point optimizer.

- `MSK_DPAR_INTPNT_NL_TOL_DFEAS` 242
  Dual feasibility tolerance used when a nonlinear model is solved.

- MSK_DPAR_INTPNT_NL_TOL_MU_RED 242
  Relative complementarity gap tolerance.

- MSK_DPAR_INTPNT_NL_TOL_NEAR_REL 242
  Nonlinear solver optimality tolerance parameter.

- MSK_DPAR_INTPNT_NL_TOL_PFEAS 243
  Primal feasibility tolerance used when a nonlinear model is solved.

- MSK_DPAR_INTPNT_NL_TOL_REL_GAP 243
  Relative gap termination tolerance for nonlinear problems.

- MSK_DPAR_INTPNT_TOL_DFEAS 244
  Dual feasibility tolerance used for linear and quadratic optimization problems.

- MSK_DPAR_INTPNT_TOL_INFEAS 244
  Nonlinear solver infeasibility tolerance parameter.

- MSK_DPAR_INTPNT_TOL_MU_RED 244
  Relative complementarity gap tolerance.

- MSK_DPAR_INTPNT_TOL_PFEAS 245
  Primal feasibility tolerance used for linear and quadratic optimization problems.

- MSK_DPAR_INTPNT_TOL_REL_GAP 246
  Relative gap termination tolerance.

- MSK_DPAR_LOWER_OBJ_CUT 246
  Objective bound.

- MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH 247
  Objective bound.

- MSK_DPAR_MIO_DISABLE_TERM_TIME 247
  Certain termination criterias is disabled within the mixed integer optimizer for period time specified by the parameter.

- MSK_IPAR_MIO_MAX_NUM_BRANCHES 294
  Maximum number of branches allowed during the branch and bound search.

- MSK_IPAR_MIO_MAX_NUM_SOLUTIONS 295
  Controls how many feasible solutions the mixed-integer optimizer investigates.

- MSK_DPAR_MIO_MAX_TIME 248
  Time limit for the mixed integer optimizer.

- MSK_DPAR_MIO_NEAR_TOL_REL_GAP 249
  The mixed integer optimizer is terminated when this tolerance is satisfied.

- MSK_DPAR_MIO_TOL_REL_GAP 250
  Relative optimality tolerance employed by the mixed integer optimizer.

- MSK_DPAR_OPTIMIZER_MAX_TIME 252
  Solver time limit.

- MSK_IPAR_SIM_MAX_ITERATIONS 316
  Maximum number of iterations that can be used by a simplex optimizer.

- MSK_DPAR_UPPER_OBJ_CUT 253
  Objective bound.

- MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH 254
  Objective bound.

C.1.13 Progress call-back parameters.

- MSK_DPAR_CALLBACK_FREQ 237
  Controls progress call-back frequency.

- MSK_IPAR_SOLUTION_CALLBACK 323
  Indicates whether solution call-backs will be performed during the optimization.

C.1.14 Non-convex solver parameters.

- MSK_IPAR_LOG_NONCONVEX 286
  Controls amount of output printed by the nonconvex optimizer.

- MSK_IPAR_NONCONVEX_MAX_ITERATIONS 299
  Maximum number of iterations that can be used by the nonconvex optimizer.

- MSK_DPAR_NONCONVEX_TOL_FEAS 251
  Feasibility tolerance used by the nonconvex optimizer.

- MSK_DPAR_NONCONVEX_TOL_OPT 251
  Optimality tolerance used by the nonconvex optimizer.

C.1.15 Feasibility repair parameters.

- MSK_DPAR_FEASREPAIR_TOL 239
  Tolerance for constraint enforcing upper bound on sum of weighted violations in feasibility repair.

C.1.16 Optimization system parameters.

Parameters defining the overall solver system environment. This includes system and platform related information and behavior.

- MSK_IPAR_CACHE_SIZE_L1 268
  Specifies the size of the level 1 cache of the processor.
- MSK_IPAR_CACHE_SIZE_L2 269
  Specifies the size of the level 2 cache of the processor.
- MSK_IPAR_CHECK_CTRL_C 269
  Turns ctrl-c check on or off.
- MSK_IPAR_CPU_TYPE 272
  Specifies the CPU type.
- MSK_IPAR_INTPNT_NUM_THREADS 277
  Controls the number of threads employed by the interior-point optimizer.
- MSK_IPAR_LICENSE_CACHE_TIME 280
  Controls the license manager client behavior.
- MSK_IPAR_LICENSE_CHECK_TIME 280
  Controls the license manager client behavior.
- MSK_IPAR_LICENSE_WAIT 281
  Controls if MOSEK should queue for a license if none is available.
- MSK_IPAR_LOG_STORAGE 289
  Controls the memory related log information.

C.1.17 Output information parameters.

- MSK_IPAR_FLUSH_STREAM_FREQ 273
  Controls the stream flushing frequency.
- MSK_IPAR_INFEAS_REPORT_LEVEL 274
  Controls the contents of the infeasibility report.
- MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS 281
  Controls license manager client behavior.
- MSK_IPAR_LOG 282
  Controls the amount of log information.

- MSK_IPAR_LOG_BI 282
  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 282
  Controls the logging frequency.

- MSK_IPAR_LOG_CUT_SECOND_OPT 283
  Controls the reduction in the log levels for the second and any subsequent optimizations.

- MSK_IPAR_LOG_FACTOR 283
  If turned on, then the factor log lines are added to the log.

- MSK_IPAR_LOG_FEASREPAIR 284
  Controls the amount of output printed when performing feasibility repair.

- MSK_IPAR_LOG_FILE 284
  If turned on, then some log info is printed when a file is written or read.

- MSK_IPAR_LOG_HEAD 284
  If turned on, then a header line is added to the log.

- MSK_IPAR_LOG_INFEAS_ANA 285
  Controls log level for the infeasibility analyzer.

- MSK_IPAR_LOG_INTPNT 285
  Controls the amount of log information from the interior-point optimizers.

- MSK_IPAR_LOG_MIO 285
  Controls the amount of log information from the mixed-integer optimizers.

- MSK_IPAR_LOG_MIO_FREQ 285
  The mixed integer solver logging frequency.

- MSK_IPAR_LOG_NONCONVEX 286
  Controls amount of output printed by the nonconvex optimizer.

- MSK_IPAR_LOG_OPTIMIZER 286
  Controls the amount of general optimizer information that is logged.

- MSK_IPAR_LOG_ORDER 286
  If turned on, then factor lines are added to the log.

- MSK_IPAR_LOG_PARAM 287
  Controls the amount of information printed out about parameter changes.

- `MSK_IPAR_LOG_RESPONSE` 287
  Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.

- `MSK_IPAR_LOG_SENSITIVITY` 287
  Control logging in sensitivity analyzer.

- `MSK_IPAR_LOG_SENSITIVITY_OPT` 288
  Control logging in sensitivity analyzer.

- `MSK_IPAR_LOG_SIM` 288
  Controls the amount of log information from the simplex optimizers.

- `MSK_IPAR_LOG_SIM_FREQ` 288
  Controls simplex logging frequency.

- `MSK_IPAR_LOG_SIM_MINOR` 289
  Currently not in use.

- `MSK_IPAR_LOG_SIM_NETWORK_FREQ` 289
  Controls the network simplex logging frequency.

- `MSK_IPAR_LOG_STORAGE` 289
  Controls the memory related log information.

- `MSK_IPAR_MAX_NUM_WARNINGS` 290
  Waning level. A higher value results in more warnings.

- `MSK_IPAR_WARNING_LEVEL` 323
  Warning level.

C.1.18 Extra information about the optimization problem.

- `MSK_IPAR_OBJECTIVE_SENSE` 299
  If the objective sense for the task is undefined, then the value of this parameter is used as the default objective sense.

C.1.19 Overall solver parameters.

- `MSK_IPAR_BI_CLEAN_OPTIMIZER` 267
  Controls which simplex optimizer is used in the clean-up phase.

- `MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS` 270
  The maximum number of simultaneous optimizations that will be started by the concurrent optimizer.

- `MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX` . . . 270
  Priority of the dual simplex algorithm when selecting solvers for concurrent optimization.

- `MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX` . . . 271
  Priority of the free simplex optimizer when selecting solvers for concurrent optimization.

- `MSK_IPAR_CONCURRENT_PRIORITY_INTPNT` . . . 271
  Priority of the interior-point algorithm when selecting solvers for concurrent optimization.

- `MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX` . . . 271
  Priority of the primal simplex algorithm when selecting solvers for concurrent optimization.

- `MSK_IPAR_DATA_CHECK` . . . 272
  Enable data checking for debug purposes.

- `MSK_IPAR_FEASREPAIR_OPTIMIZE` . . . 273
  Controls which type of feasibility analysis is to be performed.

- `MSK_IPAR_INFEAS_PREFER_PRIMAL` . . . 274
  Controls which certificate is used if both primal- and dual- certificate of infeasibility is available.

- `MSK_IPAR_LICENSE_WAIT` . . . 281
  Controls if MOSEK should queue for a license if none is available.

- `MSK_IPAR_MIO_CONT_SOL` . . . 291
  Controls the meaning of interior-point and basic solutions in MIP problems.

- `MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER` . . . 294
  Controls the size of the local search space when doing local branching.

- `MSK_IPAR_MIO_MODE` . . . 295
  Turns on/off the mixed integer mode.

- `MSK_IPAR_OPTIMIZER` . . . 302
  Controls which optimizer is used to optimize the task.

- `MSK_IPAR_PRESOLVE_LEVEL` . . . 304
  Currently not used.

- `MSK_IPAR_PRESOLVE_USE` . . . 305
  Controls whether the presolve is applied to a problem before it is optimized.

- `MSK_IPAR_SENSITIVITY_OPTIMIZER` 313
  Controls which optimizer is used for optimal partition sensitivity analysis.

- `MSK_IPAR_SENSITIVITY_TYPE` 313
  Controls which type of sensitivity analysis is to be performed.

- `MSK_IPAR_SOLUTION_CALLBACK` 323
  Indicates whether solution call-backs will be performed during the optimization.

C.1.20 Behavior of the optimization task.

Parameters defining the behavior of an optimization task when loading data.

- `MSK_IPAR_ALLOC_ADD_QNZ` 267
  Controls how the quadratic matrixes are extended.

- `MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL` 334
  Feasibility repair name violation name.

- `MSK_IPAR_MAXNUMANZ_DOUBLE_TRH` 290
  Controls how the constraint matrix is extended.

- `MSK_IPAR_READ_ADD_ANZ` 305
  Controls how the constraint matrix is extended.

- `MSK_IPAR_READ_ADD_CON` 306
  Additional number of constraints that is made room for in the problem.

- `MSK_IPAR_READ_ADD_CONE` 306
  Additional number of conic constraints that is made room for in the problem.

- `MSK_IPAR_READ_ADD_QNZ` 306
  Controls how the quadratic matrixes are extended.

- `MSK_IPAR_READ_ADD_VAR` 306
  Additional number of variables that is made room for in the problem.

- `MSK_IPAR_READ_ANZ` 307
  Controls the expected number of constraint non-zeros.

- `MSK_IPAR_READ_CON` 307
  Controls the expected number of constraints.

- `MSK_IPAR_READ_CONE` 307
  Controls the expected number of conic constraints.

- MSK_IPAR_READ_QNZ 312
  Controls the expected number of quadratic non-zeros.

- MSK_IPAR_READ_TASK_IGNORE_PARAM 312
  Controls what information is used from the task files.

- MSK_IPAR_READ_VAR 312
  Controls the expected number of variables.

- MSK_IPAR_WRITE_TASK_INC_SOL 331
  Controls whether the solutions are stored in the task file too.

C.1.21 Data input/output parameters.

Parameters defining the behavior of data readers and writers.

- MSK_SPAR_BAS_SOL_FILE_NAME 333
  Name of the bas solution file.

- MSK_SPAR_DATA_FILE_NAME 333
  Data are read and written to this file.

- MSK_SPAR_DEBUG_FILE_NAME 334
  MOSEK debug file.

- MSK_IPAR_INFEAS_REPORT_AUTO 274
  Turns the feasibility report on or off.

- MSK_SPAR_INT_SOL_FILE_NAME 334
  Name of the int solution file.

- MSK_SPAR_ITR_SOL_FILE_NAME 334
  Name of the itr solution file.

- MSK_IPAR_LOG_FILE 284
  If turned on, then some log info is printed when a file is written or read.

- MSK_IPAR_LP_WRITE_IGNORE_INCOMPATIBLE_ITEMS 290
  Controls the result of writing a problem containing incompatible items to an LP file.

- MSK_IPAR_OPF_MAX_TERMS_PER_LINE 299
  The maximum number of terms (linear and quadratic) per line when an OPF file is written.

- MSK_IPAR_OPF_WRITE_HEADER 300
  Write a text header with date and MOSEK version in an OPF file.

- `MSK_IPAR_OPF_WRITE_HINTS` 300
  Write a hint section with problem dimensions in the beginning of an OPF file.

- `MSK_IPAR_OPF_WRITE_PARAMETERS` 300
  Write a parameter section in an OPF file.

- `MSK_IPAR_OPF_WRITE_PROBLEM` 301
  Write objective, constraints, bounds etc. to an OPF file.

- `MSK_IPAR_OPF_WRITE_SOL_BAS` 301
  Controls what is written to the OPF files.

- `MSK_IPAR_OPF_WRITE_SOL_ITG` 301
  Controls what is written to the OPF files.

- `MSK_IPAR_OPF_WRITE_SOL_ITR` 302
  Controls what is written to the OPF files.

- `MSK_IPAR_OPF_WRITE_SOLUTIONS` 302
  Enable inclusion of solutions in the OPF files.

- `MSK_SPAR_PARAM_COMMENT_SIGN` 335
  Solution file comment character.

- `MSK_IPAR_PARAM_READ_CASE_NAME` 303
  If turned on, then names in the parameter file are case sensitive.

- `MSK_SPAR_PARAM_READ_FILE_NAME` 335
  Modifications to the parameter database is read from this file.

- `MSK_IPAR_PARAM_READ_IGN_ERROR` 303
  If turned on, then errors in paramter settings is ignored.

- `MSK_SPAR_PARAM_WRITE_FILE_NAME` 335
  The parameter database is written to this file.

- `MSK_IPAR_READ_ADD_ANZ` 305
  Controls how the constraint matrix is extended.

- `MSK_IPAR_READ_ADD_CON` 306
  Additional number of constraints that is made room for in the problem.

- `MSK_IPAR_READ_ADD_CONE` 306
  Additional number of conic constraints that is made room for in the problem.

- `MSK_IPAR_READ_ADD_QNZ` 306
  Controls how the quadratic matrixes are extended.

- `MSK_IPAR_READ_ADD_VAR` 306
  Additional number of variables that is made room for in the problem.

- `MSK_IPAR_READ_ANZ` 307
  Controls the expected number of constraint non-zeros.

- `MSK_IPAR_READ_CON` 307
  Controls the expected number of constraints.

- `MSK_IPAR_READ_CONE` 307
  Controls the expected number of conic constraints.

- `MSK_IPAR_READ_DATA_COMPRESSED` 308
  Controls the input file decompression.

- `MSK_IPAR_READ_DATA_FORMAT` 308
  Format of the data file to be read.

- `MSK_IPAR_READ_KEEP_FREE_CON` 308
  Controls whether the free constraints are included in the problem.

- `MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU` 309
  Controls how the LP files are interpreted.

- `MSK_IPAR_READ_LP_QUOTED_NAMES` 309
  If a name is in quotes when reading an LP file, the quotes will be removed.

- `MSK_SPAR_READ_MPS_BOU_NAME` 335
  Name of the BOUNDS vector used. An empty name means that the first BOUNDS vector is used.

- `MSK_IPAR_READ_MPS_FORMAT` 309
  Controls how strictly the MPS file reader interprets the MPS format.

- `MSK_IPAR_READ_MPS_KEEP_INT` 310
  Controls if integer constraints are read.

- `MSK_SPAR_READ_MPS_OBJ_NAME` 336
  Objective name in the MPS file.

- `MSK_IPAR_READ_MPS_OBJ_SENSE` 310
  Controls the MPS format extensions.

- `MSK_IPAR_READ_MPS_QUOTED_NAMES` 310
  Controls the MPS format extensions.

- `MSK_SPAR_READ_MPS_RAN_NAME` 336
  Name of the RANGE vector used. An empty name means that the first RANGE vector is used.

- `MSK_IPAR_READ_MPS_RELAX` 311
  Controls the meaning of integer constraints.

- `MSK_SPAR_READ_MPS_RHS_NAME` 336
  Name of the RHS used. An empty name means that the first RHS vector is used.

- `MSK_IPAR_READ_MPS_WIDTH` 311
  Controls the maximal number of chars allowed in one line of the MPS file.

- `MSK_IPAR_READ_Q_MODE` 311
  Controls how the Q matrices are read from the MPS file.

- `MSK_IPAR_READ_QNZ` 312
  Controls the expected number of quadratic non-zeros.

- `MSK_IPAR_READ_TASK_IGNORE_PARAM` 312
  Controls what information is used from the task files.

- `MSK_IPAR_READ_VAR` 312
  Controls the expected number of variables.

- `MSK_SPAR_SOL_FILTER_XC_LOW` 337
  Solution file filter.

- `MSK_SPAR_SOL_FILTER_XC_UPR` 337
  Solution file filter.

- `MSK_SPAR_SOL_FILTER_XX_LOW` 337
  Solution file filter.

- `MSK_SPAR_SOL_FILTER_XX_UPR` 338
  Solution file filter.

- `MSK_IPAR_SOL_QUOTED_NAMES` 322
  Controls the solution file format.

- `MSK_IPAR_SOL_READ_NAME_WIDTH` 322
  Controls the input solution file format.

- `MSK_IPAR_SOL_READ_WIDTH` 322
  Controls the input solution file format.

- `MSK_SPAR_STAT_FILE_NAME` 338
  Statistics file name.
- `MSK_SPAR_STAT_KEY` 338
  Key used when writing the summary file.
- `MSK_SPAR_STAT_NAME` 339
  Name used when writing the statistics file.
- `MSK_IPAR_WRITE_BAS_CONSTRAINTS` 323
  Controls the basic solution file format.
- `MSK_IPAR_WRITE_BAS_HEAD` 324
  Controls the basic solution file format.
- `MSK_IPAR_WRITE_BAS_VARIABLES` 324
  Controls the basic solution file format.
- `MSK_IPAR_WRITE_DATA_COMPRESSED` 324
  Controls output file compression.
- `MSK_IPAR_WRITE_DATA_FORMAT` 325
  Controls the output file problem format.
- `MSK_IPAR_WRITE_DATA_PARAM` 325
  Controls output file data.
- `MSK_IPAR_WRITE_FREE_CON` 325
  Controls the output file data.
- `MSK_IPAR_WRITE_GENERIC_NAMES` 326
  Controls the output file data.
- `MSK_IPAR_WRITE_GENERIC_NAMES_IO` 326
  Index origin used in generic names.
- `MSK_IPAR_WRITE_INT_CONSTRAINTS` 326
  Controls the integer solution file format.
- `MSK_IPAR_WRITE_INT_HEAD` 327
  Controls the integer solution file format.
- `MSK_IPAR_WRITE_INT_VARIABLES` 327
  Controls the integer solution file format.
- `MSK_SPAR_WRITE_LP_GEN_VAR_NAME` 339
  Added variable names in the LP files.

- `MSK_IPAR_WRITE_LP_LINE_WIDTH` 327
  Controls the LP output file format.

- `MSK_IPAR_WRITE_LP_QUOTED_NAMES` 327
  Controls LP output file format.

- `MSK_IPAR_WRITE_LP_STRICT_FORMAT` 328
  Controls whether LP output files satisfy the LP format strictly.

- `MSK_IPAR_WRITE_LP_TERMS_PER_LINE` 328
  Controls the LP output file format.

- `MSK_IPAR_WRITE_MPS_INT` 328
  Controls the output file data.

- `MSK_IPAR_WRITE_MPS_OBJ_SENSE` 329
  Controls the output file data.

- `MSK_IPAR_WRITE_MPS_QUOTED_NAMES` 329
  Controls the output file data.

- `MSK_IPAR_WRITE_MPS_STRICT` 329
  Controls the output MPS file format.

- `MSK_IPAR_WRITE_PRECISION` 330
  Controls data precision employed in when writing an MPS file.

- `MSK_IPAR_WRITE_SOL_CONSTRAINTS` 330
  Controls the solution file format.

- `MSK_IPAR_WRITE_SOL_HEAD` 330
  Controls solution file format.

- `MSK_IPAR_WRITE_SOL_VARIABLES` 331
  Controls the solution file format.

- `MSK_IPAR_WRITE_TASK_INC_SOL` 331
  Controls whether the solutions are stored in the task file too.

- `MSK_IPAR_WRITE_XML_MODE` 331
  Controls if linear coefficients should be written by row or column when writing in the XML file format.

C.1.22 Solution input/output parameters.

Parameters defining the behavior of solution reader and writer.

- MSK_SPAR_BAS_SOL_FILE_NAME 333
  Name of the bas solution file.
- MSK_IPAR_INFEAS_REPORT_AUTO 274
  Turns the feasibility report on or off.
- MSK_SPAR_INT_SOL_FILE_NAME 334
  Name of the int solution file.
- MSK_SPAR_ITR_SOL_FILE_NAME 334
  Name of the itr solution file.
- MSK_IPAR_SOL_FILTER_KEEP_BASIC 321
  Controls the license manager client behavior.
- MSK_IPAR_SOL_FILTER_KEEP_RANGED 321
  Control the contents of the solution files.
- MSK_SPAR_SOL_FILTER_XC_LOW 337
  Solution file filter.
- MSK_SPAR_SOL_FILTER_XC_UPR 337
  Solution file filter.
- MSK_SPAR_SOL_FILTER_XX_LOW 337
  Solution file filter.
- MSK_SPAR_SOL_FILTER_XX_UPR 338
  Solution file filter.
- MSK_IPAR_SOL_QUOTED_NAMES 322
  Controls the solution file format.
- MSK_IPAR_SOL_READ_NAME_WIDTH 322
  Controls the input solution file format.
- MSK_IPAR_SOL_READ_WIDTH 322
  Controls the input solution file format.
- MSK_IPAR_WRITE_BAS_CONSTRAINTS 323
  Controls the basic solution file format.

- `MSK_IPAR_WRITE_BAS_HEAD` 324
  Controls the basic solution file format.

- `MSK_IPAR_WRITE_BAS_VARIABLES` 324
  Controls the basic solution file format.

- `MSK_IPAR_WRITE_INT_CONSTRAINTS` 326
  Controls the integer solution file format.

- `MSK_IPAR_WRITE_INT_HEAD` 327
  Controls the integer solution file format.

- `MSK_IPAR_WRITE_INT_VARIABLES` 327
  Controls the integer solution file format.

- `MSK_IPAR_WRITE_SOL_CONSTRAINTS` 330
  Controls the solution file format.

- `MSK_IPAR_WRITE_SOL_HEAD` 330
  Controls solution file format.

- `MSK_IPAR_WRITE_SOL_VARIABLES` 331
  Controls the solution file format.

C.1.23 Infeasibility report parameters.

- `MSK_IPAR_INFEAS_GENERIC_NAMES` 273
  Controls the contents of the infeasibility report.

- `MSK_IPAR_INFEAS_REPORT_LEVEL` 274
  Controls the contents of the infeasibility report.

- `MSK_IPAR_LOG_INFEAS_ANA` 285
  Controls log level for the infeasibility analyzer.

C.1.24 License manager parameters.

- `MSK_IPAR_LICENSE_ALLOW_OVERUSE` 279
  Controls if license overuse is allowed when caching licenses

- `MSK_IPAR_LICENSE_CACHE_TIME` 280
  Controls the license manager client behavior.

- `MSK_IPAR_LICENSE_CHECK_TIME` 280
  Controls the license manager client behavior.

- `MSK_IPAR_LICENSE_DEBUG` 280
  Controls the license manager client debugging behavior.
- `MSK_IPAR_LICENSE_PAUSE_TIME` 281
  Controls license manager client behavior.
- `MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS` 281
  Controls license manager client behavior.
- `MSK_IPAR_LICENSE_WAIT` 281
  Controls if MOSEK should queue for a license if none is available.

C.1.25 Data check parameters.

These parameters defines data checking settings and problem data tolerances, i.e. which values are rounded to 0 or infinity, and which values are large or small enough to produce a warning.

- `MSK_IPAR_CHECK_CONVEXITY` 269
  Specify the level of convexity check on quadratic problems
- `MSK_IPAR_CHECK_TASK_DATA` 270
  If this feature is turned on, then the task data is checked for bad values i.e. NaNs. before an optimization is performed.
- `MSK_DPAR_DATA_TOL_AIJ` 237
  Data tolerance threshold.
- `MSK_DPAR_DATA_TOL_AIJ_LARGE` 237
  Data tolerance threshold.
- `MSK_DPAR_DATA_TOL_BOUND_INF` 238
  Data tolerance threshold.
- `MSK_DPAR_DATA_TOL_BOUND_WRN` 238
  Data tolerance threshold.
- `MSK_DPAR_DATA_TOL_C_HUGE` 238
  Data tolerance threshold.
- `MSK_DPAR_DATA_TOL_CJ_LARGE` 239
  Data tolerance threshold.
- `MSK_DPAR_DATA_TOL_QIJ` 239
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_X 239
  Data tolerance threshold.

C.2 Double parameters

- MSK_DPAR_BASIS_REL_TOL_S 236
  Maximum relative dual bound violation allowed in an optimal basic solution.

- MSK_DPAR_BASIS_TOL_S 236
  Maximum absolute dual bound violation in an optimal basic solution.

- MSK_DPAR_BASIS_TOL_X 236
  Maximum absolute primal bound violation allowed in an optimal basic solution.

- MSK_DPAR_BI_LU_TOL_REL_PIV 236
  Relative pivot tolerance used in the LU factorization in the basis identification procedure.

- MSK_DPAR_CALLBACK_FREQ 237
  Controls progress call-back frequency.

- MSK_DPAR_DATA_TOL_AIJ 237
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_AIJ_LARGE 237
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_BOUND_INF 238
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_BOUND_WRN 238
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_C_HUGE 238
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_CJ_LARGE 239
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_QIJ 239
  Data tolerance threshold.

- MSK_DPAR_DATA_TOL_X 239
  Data tolerance threshold.

- `MSK_DPAR_FEASREPAIR_TOL` 239
  Tolerance for constraint enforcing upper bound on sum of weighted violations in feasibility repair.

- `MSK_DPAR_INTPNT_CO_TOL_DFEAS` 240
  Dual feasibility tolerance used by the conic interior-point optimizer.

- `MSK_DPAR_INTPNT_CO_TOL_INFEAS` 240
  Infeasibility tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_MU_RED` 240
  Optimality tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_NEAR_REL` 241
  Optimality tolerance for the conic solver.

- `MSK_DPAR_INTPNT_CO_TOL_PFEAS` 241
  Primal feasibility tolerance used by the conic interior-point optimizer.

- `MSK_DPAR_INTPNT_CO_TOL_REL_GAP` 241
  Relative gap termination tolerance used by the conic interior-point optimizer.

- `MSK_DPAR_INTPNT_NL_MERIT_BAL` 242
  Controls if the complementarity and infeasibility is converging to zero at about equal rates.

- `MSK_DPAR_INTPNT_NL_TOL_DFEAS` 242
  Dual feasibility tolerance used when a nonlinear model is solved.

- `MSK_DPAR_INTPNT_NL_TOL_MU_RED` 242
  Relative complementarity gap tolerance.

- `MSK_DPAR_INTPNT_NL_TOL_NEAR_REL` 242
  Nonlinear solver optimality tolerance parameter.

- `MSK_DPAR_INTPNT_NL_TOL_PFEAS` 243
  Primal feasibility tolerance used when a nonlinear model is solved.

- `MSK_DPAR_INTPNT_NL_TOL_REL_GAP` 243
  Relative gap termination tolerance for nonlinear problems.

- `MSK_DPAR_INTPNT_NL_TOL_REL_STEP` 243
  Relative step size to the boundary for general nonlinear optimization problems.

- `MSK_DPAR_INTPNT_TOL_DFEAS` 244
  Dual feasibility tolerance used for linear and quadratic optimization problems.

- `MSK_DPAR_INTPNT_TOL_DSAFE` ... 244
  Controls the interior-point dual starting point.

- `MSK_DPAR_INTPNT_TOL_INFEAS` ... 244
  Nonlinear solver infeasibility tolerance parameter.

- `MSK_DPAR_INTPNT_TOL_MU_RED` ... 244
  Relative complementarity gap tolerance.

- `MSK_DPAR_INTPNT_TOL_PATH` ... 245
  interior-point centering aggressiveness.

- `MSK_DPAR_INTPNT_TOL_PFEAS` ... 245
  Primal feasibility tolerance used for linear and quadratic optimization problems.

- `MSK_DPAR_INTPNT_TOL_PSAFE` ... 245
  Controls the interior-point primal starting point.

- `MSK_DPAR_INTPNT_TOL_REL_GAP` ... 246
  Relative gap termination tolerance.

- `MSK_DPAR_INTPNT_TOL_REL_STEP` ... 246
  Relative step size to the boundary for linear and quadratic optimization problems.

- `MSK_DPAR_INTPNT_TOL_STEP_SIZE` ... 246
  If the step size falls below the value of this parameter, then the interior-point optimizer assumes it is stalled. It it does not not make any progress.

- `MSK_DPAR_LOWER_OBJ_CUT` ... 246
  Objective bound.

- `MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH` ... 247
  Objective bound.

- `MSK_DPAR_MIO_DISABLE_TERM_TIME` ... 247
  Certain termination criterias is disabled within the mixed integer optimizer for period time specified by the parameter.

- `MSK_DPAR_MIO_HEURISTIC_TIME` ... 248
  Time limit for the mixed integer heuristics.

- `MSK_DPAR_MIO_MAX_TIME` ... 248
  Time limit for the mixed integer optimizer.

- `MSK_DPAR_MIO_MAX_TIME_APRX_OPT` ... 248
  Time limit for the mixed integer optimizer.

- `MSK_DPAR_MIO_NEAR_TOL_ABS_GAP` 249
  Relaxed absolute optimality tolerance employed by the mixed integer optimizer.

- `MSK_DPAR_MIO_NEAR_TOL_REL_GAP` 249
  The mixed integer optimizer is terminated when this tolerance is satisfied.

- `MSK_DPAR_MIO_REL_ADD_CUT_LIMITED` 249
  Controls cut generation for MIP solver.

- `MSK_DPAR_MIO_TOL_ABS_GAP` 250
  Absolute optimality tolerance employed by the mixed integer optimizer.

- `MSK_DPAR_MIO_TOL_ABS_RELAX_INT` 250
  Integer constraint tolerance.

- `MSK_DPAR_MIO_TOL_REL_GAP` 250
  Relative optimality tolerance employed by the mixed integer optimizer.

- `MSK_DPAR_MIO_TOL_REL_RELAX_INT` 251
  Integer constraint tolerance.

- `MSK_DPAR_MIO_TOL_X` 251
  Absolute solution tolerance used in mixed-integer optimizer.

- `MSK_DPAR_NONCONVEX_TOL_FEAS` 251
  Feasibility tolerance used by the nonconvex optimizer.

- `MSK_DPAR_NONCONVEX_TOL_OPT` 251
  Optimality tolerance used by the nonconvex optimizer.

- `MSK_DPAR_OPTIMIZER_MAX_TIME` 252
  Solver time limit.

- `MSK_DPAR_PRESOLVE_TOL_AIJ` 252
  Absolute zero tolerance employed for constraint coefficients in the presolve.

- `MSK_DPAR_PRESOLVE_TOL_LIN_DEP` 252
  Controls when a constraint is determined to be linearly dependent.

- `MSK_DPAR_PRESOLVE_TOL_S` 253
  Absolute zero tolerance employed for slack variables in the presolve.

- `MSK_DPAR_PRESOLVE_TOL_X` 253
  Absolute zero tolerance employed for variables in the presolve.

- `MSK_DPAR_SIMPLEX_ABS_TOL_PIV` 253
  Absolute pivot tolerance employed by the simplex optimizers.

- `MSK_DPAR_UPPER_OBJ_CUT` 253
  Objective bound.

- `MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH` 254
  Objective bound.

- `basis_rel_tol_s`

  Corresponding constant:
  `MSK_DPAR_BASIS_REL_TOL_S`

  Description:
  Maximum relative dual bound violation allowed in an optimal basic solution.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-12

- `basis_tol_s`

  Corresponding constant:
  `MSK_DPAR_BASIS_TOL_S`

  Description:
  Maximum absolute dual bound violation in an optimal basic solution.

  Possible Values:
  Any number between 1.0e-9 and +inf.

  Default value:
  1.0e-6

- `basis_tol_x`

  Corresponding constant:
  `MSK_DPAR_BASIS_TOL_X`

  Description:
  Maximum absolute primal bound violation allowed in an optimal basic solution.

  Possible Values:
  Any number between 1.0e-9 and +inf.

  Default value:
  1.0e-6

- `bi_lu_tol_rel_piv`

Corresponding constant:

`MSK_DPAR_BI_LU_TOL_REL_PIV`

Description:

Relative pivot tolerance used in the LU factorization in the basis identification procedure.

Possible Values:

Any number between 1.0e-6 and 0.999999.

Default value:

0.01

- `callback_freq`

  Corresponding constant:

  `MSK_DPAR_CALLBACK_FREQ`

  Description:

  Controls the time between calls to the progress call-back function. Hence, if the value of this parameter is for example 10, then the call-back is called approximately each 10 seconds. A negative value is equivalent to infinity.

  In general frequent call-backs may hurt the performance.

  Possible Values:

  Any number between -inf and +inf.

  Default value:

  -1.0

- `data_tol_aij`

  Corresponding constant:

  `MSK_DPAR_DATA_TOL_AIJ`

  Description:

  Absolute zero tolerance for elements in A.

  Possible Values:

  Any number between 1.0e-16 and 1.0e-6.

  Default value:

  1.0e-12

- `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

- `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

- `data_tol_bound_wrn`

  Corresponding constant:
  `MSK_DPAR_DATA_TOL_BOUND_WRN`

  Description:
  If a bound value is larger than this value in absolute size, then a warning message is issued.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e8

- `data_tol_c_huge`

  Corresponding constant:
  `MSK_DPAR_DATA_TOL_C_HUGE`

  Description:
  An element in c which is larger than the value of this parameter in absolute terms is considered to be huge and generates an error.

  Possible Values:
  Any number between 0.0 and +inf.

Default value:
1.0e16

- `data_tol_cj_large`

  Corresponding constant:
  `MSK_DPAR_DATA_TOL_CJ_LARGE`

  Description:
  An element in c which is larger than this value in absolute terms causes a warning message to be printed.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e8

- `data_tol_qij`

  Corresponding constant:
  `MSK_DPAR_DATA_TOL_QIJ`

  Description:
  Absolute zero tolerance for elements in Q matrices.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-16

- `data_tol_x`

  Corresponding constant:
  `MSK_DPAR_DATA_TOL_X`

  Description:
  Zero tolerance for constraints and variables i.e. if the distance between the lower and upper bound is less than this value, then the lower and lower bound is considered identical.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-8

- `feasrepair_tol`

Corresponding constant:
`MSK_DPAR_FEASREPAIR_TOL`

Description:
Tolerance for constraint enforcing upper bound on sum of weighted violations in feasibility repair.

Possible Values:
Any number between 1.0e-16 and 1.0e+16.

Default value:
1.0e-10

- `intpnt_co_tol_dfeas`

  Corresponding constant:
  `MSK_DPAR_INTPNT_CO_TOL_DFEAS`

  Description:
  Dual feasibility tolerance used by the conic interior-point optimizer.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-8

- `intpnt_co_tol_infeas`

  Corresponding constant:
  `MSK_DPAR_INTPNT_CO_TOL_INFEAS`

  Description:
  Controls when the conic interior-point optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-8

- `intpnt_co_tol_mu_red`

  Corresponding constant:
  `MSK_DPAR_INTPNT_CO_TOL_MU_RED`

  Description:
  Relative complementarity gap tolerance feasibility tolerance used by the conic interior-point optimizer.

Possible Values:
Any number between 0.0 and 1.0.

Default value:
1.0e-8

- `intpnt_co_tol_near_rel`

  Corresponding constant:
  `MSK_DPAR_INTPNT_CO_TOL_NEAR_REL`

  Description:
  If MOSEK cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.

  Possible Values:
  Any number between 1.0 and +inf.

  Default value:
  100

- `intpnt_co_tol_pfeas`

  Corresponding constant:
  `MSK_DPAR_INTPNT_CO_TOL_PFEAS`

  Description:
  Primal feasibility tolerance used by the conic interior-point optimizer.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-8

- `intpnt_co_tol_rel_gap`

  Corresponding constant:
  `MSK_DPAR_INTPNT_CO_TOL_REL_GAP`

  Description:
  Relative gap termination tolerance used by the conic interior-point optimizer.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-8

- `intpnt_nl_merit_bal`

  Corresponding constant:
  `MSK_DPAR_INTPNT_NL_MERIT_BAL`

  Description:
  Controls if the complementarity and infeasibility is converging to zero at about equal rates.

  Possible Values:
  Any number between 0.0 and 0.99.

  Default value:
  1.0e-4

- `intpnt_nl_tol_dfeas`

  Corresponding constant:
  `MSK_DPAR_INTPNT_NL_TOL_DFEAS`

  Description:
  Dual feasibility tolerance used when a nonlinear model is solved.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-8

- `intpnt_nl_tol_mu_red`

  Corresponding constant:
  `MSK_DPAR_INTPNT_NL_TOL_MU_RED`

  Description:
  Relative complementarity gap tolerance.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-12

- `intpnt_nl_tol_near_rel`

  Corresponding constant:
  `MSK_DPAR_INTPNT_NL_TOL_NEAR_REL`

  Description:
  If the MOSEK nonlinear interior-point optimizer cannot compute a solution that

has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.

Possible Values:
Any number between 1.0 and +inf.

Default value:
1000.0

- `intpnt_nl_tol_pfeas`

  Corresponding constant:
  `MSK_DPAR_INTPNT_NL_TOL_PFEAS`

  Description:
  Primal feasibility tolerance used when a nonlinear model is solved.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-8

- `intpnt_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

- `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

- `intpnt_tol_dfeas`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_DFEAS`

  Description:
  Dual feasibility tolerance used for linear and quadratic optimization problems.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-8

- `intpnt_tol_dsafe`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_DSAFE`

  Description:
  Controls the initial dual starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly.

  Possible Values:
  Any number between 1.0e-4 and +inf.

  Default value:
  1.0

- `intpnt_tol_infeas`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_INFEAS`

  Description:
  Controls when the optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-8

- `intpnt_tol_mu_red`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_MU_RED`

Description:
Relative complementarity gap tolerance.

Possible Values:
Any number between 0.0 and 1.0.

Default value:
1.0e-16

- `intpnt_tol_path`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_PATH`

  Description:
  Controls how close the interior-point optimizer follows the central path. A large value of this parameter means the central is followed very closely. On numerical unstable problems it might worthwhile to increase this parameter.

  Possible Values:
  Any number between 0.0 and 0.9999.

  Default value:
  1.0e-8

- `intpnt_tol_pfeas`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_PFEAS`

  Description:
  Primal feasibility tolerance used for linear and quadratic optimization problems.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-8

- `intpnt_tol_psafe`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_PSAFE`

  Description:
  Controls the initial primal starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly and/or the constraint or variable bounds are very large, then it might be worthwhile to increase this value.

  Possible Values:
  Any number between 1.0e-4 and +inf.

Default value:
1.0

- `intpnt_tol_rel_gap`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_REL_GAP`

  Description:
  Relative gap termination tolerance.

  Possible Values:
  Any number between 1.0e-14 and +inf.

  Default value:
  1.0e-8

- `intpnt_tol_rel_step`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_REL_STEP`

  Description:
  Relative step size to the boundary for linear and quadratic optimization problems.

  Possible Values:
  Any number between 1.0e-4 and 0.999999.

  Default value:
  0.9999

- `intpnt_tol_step_size`

  Corresponding constant:
  `MSK_DPAR_INTPNT_TOL_STEP_SIZE`

  Description:
  If the step size falls below the value of this parameter, then the interior-point optimizer assumes it is stalled. It it does not not make any progress.

  Possible Values:
  Any number between 0.0 and 1.0.

  Default value:
  1.0e-10

- `lower_obj_cut`

  Corresponding constant:
  `MSK_DPAR_LOWER_OBJ_CUT`

Description:
If a feasible solution having an objective value 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

- `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

- `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 criterias are never checked.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  0.0

  See also:

  `MSK_IPAR_MIO_MAX_NUM_RELAXS` Maximum number of relaxations in branch and bound search.

`MSK_IPAR_MIO_MAX_NUM_BRANCHES` Maximum number of branches allowed during the branch and bound search.

`MSK_DPAR_MIO_NEAR_TOL_ABS_GAP` Relaxed absolute optimality tolerance employed by the mixed integer optimizer.

`MSK_DPAR_MIO_NEAR_TOL_REL_GAP` The mixed integer optimizer is terminated when this tolerance is satisfied.

- `mio_heuristic_time`

  Corresponding constant:
  `MSK_DPAR_MIO_HEURISTIC_TIME`

  Description:
  Minimum amount of time to be used in the heuristic search for a good feasible integer solution. A negative values implies that the optimizer decides the amount of time to be spent in the heuristic.

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1.0

- `mio_max_time`

  Corresponding constant:
  `MSK_DPAR_MIO_MAX_TIME`

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

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1.0

- `mio_max_time_aprx_opt`

  Corresponding constant:
  `MSK_DPAR_MIO_MAX_TIME_APRX_OPT`

  Description:
  Number of seconds spent by the mixed integer optimizer before the `MSK_DPAR_MIO_TOL_REL_RELAX_INT` is applied.

  Possible Values:
  Any number between 0.0 and +inf.

Default value:
60

- `mio_near_tol_abs_gap`

  Corresponding constant:
  `MSK_DPAR_MIO_NEAR_TOL_ABS_GAP`

  Description:
  Relaxed absolute optimality tolerance employed by the mixed integer optimizer. This termination criteria is delayed. See `MSK_DPAR_MIO_DISABLE_TERM_TIME` for details.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  0.0

  See also:

  `MSK_DPAR_MIO_DISABLE_TERM_TIME` Certain termination criterias is disabled within the mixed integer optimizer for period time specified by the parameter.

- `mio_near_tol_rel_gap`

  Corresponding constant:
  `MSK_DPAR_MIO_NEAR_TOL_REL_GAP`

  Description:
  The mixed integer optimizer is terminated when this tolerance is satisfied. This termination criteria is delayed. See `MSK_DPAR_MIO_DISABLE_TERM_TIME` for details.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-5

  See also:

  `MSK_DPAR_MIO_DISABLE_TERM_TIME` Certain termination criterias is disabled within the mixed integer optimizer for period time specified by the parameter.

- `mio_rel_add_cut_limited`

  Corresponding constant:
  `MSK_DPAR_MIO_REL_ADD_CUT_LIMITED`

Description:
Controls how many cuts the mixed integer optimizer is allowed to add to the problem. Let α be the value of this parameter and m the number constraints, then mixed integer optimizer is allowed to αm cuts.

Possible Values:
Any number between 0.0 and 2.0.

Default value:
0.75

- `mio_tol_abs_gap`

  Corresponding constant:
  `MSK_DPAR_MIO_TOL_ABS_GAP`

  Description:
  Absolute optimality tolerance employed by the mixed integer optimizer.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  0.0

- `mio_tol_abs_relax_int`

  Corresponding constant:
  `MSK_DPAR_MIO_TOL_ABS_RELAX_INT`

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

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-5

- `mio_tol_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-8

- `mio_tol_rel_relax_int`

  Corresponding constant:
  `MSK_DPAR_MIO_TOL_REL_RELAX_INT`

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

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-6

- `mio_tol_x`

  Corresponding constant:
  `MSK_DPAR_MIO_TOL_X`

  Description:
  Absolute solution tolerance used in mixed-integer optimizer.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-6

- `nonconvex_tol_feas`

  Corresponding constant:
  `MSK_DPAR_NONCONVEX_TOL_FEAS`

  Description:
  Feasibility tolerance used by the nonconvex optimizer.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-6

- `nonconvex_tol_opt`

  Corresponding constant:
  `MSK_DPAR_NONCONVEX_TOL_OPT`

Description:
Optimality tolerance used by the nonconvex optimizer.

Possible Values:
Any number between 0.0 and +inf.

Default value:
1.0e-7

- `optimizer_max_time`

  Corresponding constant:
  `MSK_DPAR_OPTIMIZER_MAX_TIME`

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

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1.0

- `presolve_tol_aij`

  Corresponding constant:
  `MSK_DPAR_PRESOLVE_TOL_AIJ`

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

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-12

- `presolve_tol_lin_dep`

  Corresponding constant:
  `MSK_DPAR_PRESOLVE_TOL_LIN_DEP`

  Description:
  Controls when a constraint is determined to be linearly dependent.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-6

- `presolve_tol_s`

  Corresponding constant:
  `MSK_DPAR_PRESOLVE_TOL_S`

  Description:
  Absolute zero tolerance employed for s_i in the presolve.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-8

- `presolve_tol_x`

  Corresponding constant:
  `MSK_DPAR_PRESOLVE_TOL_X`

  Description:
  Absolute zero tolerance employed for x_j in the presolve.

  Possible Values:
  Any number between 0.0 and +inf.

  Default value:
  1.0e-8

- `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

- `upper_obj_cut`

  Corresponding constant:
  `MSK_DPAR_UPPER_OBJ_CUT`

  Description:
  If a feasible solution having and objective value 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

- `upper_obj_cut_finite_trh`

Corresponding constant:
`MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH`

Description:
If the upper objective cut is greater than the value of this value parameter, then the the upper objective cut `MSK_DPAR_UPPER_OBJ_CUT` is treated as ∞.

Possible Values:
Any number between -inf and +inf.

Default value:
0.5e30

C.3 Integer parameters

- `MSK_IPAR_CHECK_CONVEXITY` 269
  Specify the level of convexity check on quadratic problems

- `MSK_IPAR_CHECK_CTRL_C` 269
  Turns ctrl-c check on or off.

- `MSK_IPAR_CHECK_TASK_DATA` 270
  If this feature is turned on, then the task data is checked for bad values i.e. NaNs. before an optimization is performed.

- `MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS` 270
  The maximum number of simultaneous optimizations that will be started by the concurrent optimizer.

- `MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX` 270
  Priority of the dual simplex algorithm when selecting solvers for concurrent optimization.

- `MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX` 271
  Priority of the free simplex optimizer when selecting solvers for concurrent optimization.

- `MSK_IPAR_CONCURRENT_PRIORITY_INTPNT` 271
  Priority of the interior-point algorithm when selecting solvers for concurrent optimization.

- `MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX` 271
  Priority of the primal simplex algorithm when selecting solvers for concurrent optimization.

- `MSK_IPAR_CPU_TYPE` 272
  Specifies the CPU type.

- `MSK_IPAR_DATA_CHECK` 272
  Enable data checking for debug purposes.

- `MSK_IPAR_FEASREPAIR_OPTIMIZE` 273
  Controls which type of feasibility analysis is to be performed.

- `MSK_IPAR_FLUSH_STREAM_FREQ` 273
  Controls the stream flushing frequency.

- `MSK_IPAR_INFEAS_GENERIC_NAMES` 273
  Controls the contents of the infeasibility report.

- `MSK_IPAR_INFEAS_PREFER_PRIMAL` 274
  Controls which certificate is used if both primal- and dual- certificate of infeasibility is available.

- `MSK_IPAR_INFEAS_REPORT_AUTO` 274
  Turns the feasibility report on or off.

- `MSK_IPAR_INFEAS_REPORT_LEVEL` 274
  Controls the contents of the infeasibility report.

- `MSK_IPAR_INTPNT_BASIS` 274
  Controls whether basis identification is performed.

- `MSK_IPAR_INTPNT_DIFF_STEP` 275
  Controls whether different step sizes are allowed in the primal and dual space.

- `MSK_IPAR_INTPNT_FACTOR_DEBUG_LVL` 275
  Controls factorization debug level.

- `MSK_IPAR_INTPNT_FACTOR_METHOD` 276
  Controls the method used to factor the Newton equation system.

- `MSK_IPAR_INTPNT_MAX_ITERATIONS` 276
  Controls the maximum number of iterations allowed in the interior-point optimizer.

- `MSK_IPAR_INTPNT_MAX_NUM_COR` 276
  Maximum number of correction steps.

- `MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS` 277
  Maximum number of steps to be used by the iterative search direction refinement.

- `MSK_IPAR_INTPNT_NUM_THREADS` 277
  Controls the number of threads employed by the interior-point optimizer.

- `MSK_IPAR_INTPNT_OFF_COL_TRH` 277
  Controls the aggressiveness of the offending column detection.

- `MSK_IPAR_INTPNT_ORDER_METHOD` 277
  Controls the ordering strategy.

- `MSK_IPAR_INTPNT_REGULARIZATION_USE` 278
  Controls whether regularization is allowed.

- `MSK_IPAR_INTPNT_SCALING` 278
  Controls how the problem is scaled before the interior-point optimizer is used.

- `MSK_IPAR_INTPNT_SOLVE_FORM` 279
  Controls whether the primal or the dual problem is solved.

- `MSK_IPAR_INTPNT_STARTING_POINT` 279
  Starting point used by the interior-point optimizer.

- `MSK_IPAR_LICENSE_ALLOW_OVERUSE` 279
  Controls if license overuse is allowed when caching licenses
- `MSK_IPAR_LICENSE_CACHE_TIME` 280
  Controls the license manager client behavior.
- `MSK_IPAR_LICENSE_CHECK_TIME` 280
  Controls the license manager client behavior.
- `MSK_IPAR_LICENSE_DEBUG` 280
  Controls the license manager client debugging behavior.
- `MSK_IPAR_LICENSE_PAUSE_TIME` 281
  Controls license manager client behavior.
- `MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS` 281
  Controls license manager client behavior.
- `MSK_IPAR_LICENSE_WAIT` 281
  Controls if MOSEK should queue for a license if none is available.
- `MSK_IPAR_LOG` 282
  Controls the amount of log information.
- `MSK_IPAR_LOG_BI` 282
  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` 282
  Controls the logging frequency.
- `MSK_IPAR_LOG_CONCURRENT` 283
  Controls amount of output printed by the concurrent optimizer.
- `MSK_IPAR_LOG_CUT_SECOND_OPT` 283
  Controls the reduction in the log levels for the second and any subsequent optimizations.
- `MSK_IPAR_LOG_FACTOR` 283
  If turned on, then the factor log lines are added to the log.
- `MSK_IPAR_LOG_FEASREPAIR` 284
  Controls the amount of output printed when performing feasibility repair.
- `MSK_IPAR_LOG_FILE` 284
  If turned on, then some log info is printed when a file is written or read.

- MSK_IPAR_LOG_HEAD 284
  If turned on, then a header line is added to the log.
- MSK_IPAR_LOG_INFEAS_ANA 285
  Controls log level for the infeasibility analyzer.
- MSK_IPAR_LOG_INTPNT 285
  Controls the amount of log information from the interior-point optimizers.
- MSK_IPAR_LOG_MIO 285
  Controls the amount of log information from the mixed-integer optimizers.
- MSK_IPAR_LOG_MIO_FREQ 285
  The mixed integer solver logging frequency.
- MSK_IPAR_LOG_NONCONVEX 286
  Controls amount of output printed by the nonconvex optimizer.
- MSK_IPAR_LOG_OPTIMIZER 286
  Controls the amount of general optimizer information that is logged.
- MSK_IPAR_LOG_ORDER 286
  If turned on, then factor lines are added to the log.
- MSK_IPAR_LOG_PARAM 287
  Controls the amount of information printed out about parameter changes.
- MSK_IPAR_LOG_PRESOLVE 287
  Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.
- MSK_IPAR_LOG_RESPONSE 287
  Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.
- MSK_IPAR_LOG_SENSITIVITY 287
  Control logging in sensitivity analyzer.
- MSK_IPAR_LOG_SENSITIVITY_OPT 288
  Control logging in sensitivity analyzer.
- MSK_IPAR_LOG_SIM 288
  Controls the amount of log information from the simplex optimizers.
- MSK_IPAR_LOG_SIM_FREQ 288
  Controls simplex logging frequency.

- MSK_IPAR_LOG_SIM_MINOR ... 289
  Currently not in use.

- MSK_IPAR_LOG_SIM_NETWORK_FREQ ... 289
  Controls the network simplex logging frequency.

- MSK_IPAR_LOG_STORAGE ... 289
  Controls the memory related log information.

- MSK_IPAR_LP_WRITE_IGNORE_INCOMPATIBLE_ITEMS ... 290
  Controls the result of writing a problem containing incompatible items to an LP file.

- MSK_IPAR_MAX_NUM_WARNINGS ... 290
  Waning level. A higher value results in more warnings.

- MSK_IPAR_MAXNUMANZ_DOUBLE_TRH ... 290
  Controls how the constraint matrix is extended.

- MSK_IPAR_MIO_BRANCH_DIR ... 290
  Controls whether the mixed integer optimizer is branching up or down by default.

- MSK_IPAR_MIO_BRANCH_PRIORITIES_USE ... 291
  Controls whether branching priorities are used by the mixed integer optimizer.

- MSK_IPAR_MIO_CONSTRUCT_SOL ... 291
  Controls if initial MIP solution should be constructed from value of integer variables.

- MSK_IPAR_MIO_CONT_SOL ... 291
  Controls the meaning of interior-point and basic solutions in MIP problems.

- MSK_IPAR_MIO_CUT_LEVEL_ROOT ... 292
  Controls the cut level employed by the mixed integer optimizer at the root node.

- MSK_IPAR_MIO_CUT_LEVEL_TREE ... 293
  Controls the cut level employed by the mixed integer optimizer in the tree.

- MSK_IPAR_MIO_FEASPUMP_LEVEL ... 293
  Controls the feasibility pump heuristic which is used to construct a good initial feasible solution.

- MSK_IPAR_MIO_HEURISTIC_LEVEL ... 293
  Controls the heuristic employed by the mixed integer optimizer to locate an initial integer feasible solution.

- MSK_IPAR_MIO_KEEP_BASIS ... 294
  Controls whether the integer presolve keeps bases in memory.

- MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER 294
  Controls the size of the local search space when doing local branching.

- MSK_IPAR_MIO_MAX_NUM_BRANCHES 294
  Maximum number of branches allowed during the branch and bound search.

- MSK_IPAR_MIO_MAX_NUM_RELAXS 295
  Maximum number of relaxations in branch and bound search.

- MSK_IPAR_MIO_MAX_NUM_SOLUTIONS 295
  Controls how many feasible solutions the mixed-integer optimizer investigates.

- MSK_IPAR_MIO_MODE 295
  Turns on/off the mixed integer mode.

- MSK_IPAR_MIO_NODE_OPTIMIZER 296
  Controls which optimizer is employed at the non-root nodes in the mixed integer optimizer.

- MSK_IPAR_MIO_NODE_SELECTION 296
  Controls the node selection strategy employed by the mixed integer optimizer.

- MSK_IPAR_MIO_PRESOLVE_AGGREGATE 297
  Controls whether problem aggregation is performed in the mixed integer presolve.

- MSK_IPAR_MIO_PRESOLVE_PROBING 297
  Controls whether probing is employed by the mixed integer presolve.

- MSK_IPAR_MIO_PRESOLVE_USE 298
  Controls whether presolve is performed by the mixed integer optimizer.

- MSK_IPAR_MIO_ROOT_OPTIMIZER 298
  Controls which optimizer is employed at the root node in the mixed integer optimizer.

- MSK_IPAR_MIO_STRONG_BRANCH 298
  The depth from the root in which strong branching is employed.

- MSK_IPAR_NONCONVEX_MAX_ITERATIONS 299
  Maximum number of iterations that can be used by the nonconvex optimizer.

- MSK_IPAR_OBJECTIVE_SENSE 299
  If the objective sense for the task is undefined, then the value of this parameter is used as the default objective sense.

- MSK_IPAR_OPF_MAX_TERMS_PER_LINE 299
  The maximum number of terms (linear and quadratic) per line when an OPF file is written.

- MSK_IPAR_OPF_WRITE_HEADER 300
  Write a text header with date and MOSEK version in an OPF file.

- MSK_IPAR_OPF_WRITE_HINTS 300
  Write a hint section with problem dimensions in the beginning of an OPF file.

- MSK_IPAR_OPF_WRITE_PARAMETERS 300
  Write a parameter section in an OPF file.

- MSK_IPAR_OPF_WRITE_PROBLEM 301
  Write objective, constraints, bounds etc. to an OPF file.

- MSK_IPAR_OPF_WRITE_SOL_BAS 301
  Controls what is written to the OPF files.

- MSK_IPAR_OPF_WRITE_SOL_ITG 301
  Controls what is written to the OPF files.

- MSK_IPAR_OPF_WRITE_SOL_ITR 302
  Controls what is written to the OPF files.

- MSK_IPAR_OPF_WRITE_SOLUTIONS 302
  Enable inclusion of solutions in the OPF files.

- MSK_IPAR_OPTIMIZER 302
  Controls which optimizer is used to optimize the task.

- MSK_IPAR_PARAM_READ_CASE_NAME 303
  If turned on, then names in the parameter file are case sensitive.

- MSK_IPAR_PARAM_READ_IGN_ERROR 303
  If turned on, then errors in paramter settings is ignored.

- MSK_IPAR_PRESOLVE_ELIM_FILL 303
  Maximum amount of fill-in in the elimination phase.

- MSK_IPAR_PRESOLVE_ELIMINATOR_USE 304
  Controls whether free or implied free variables are eliminated from the problem.

- MSK_IPAR_PRESOLVE_LEVEL 304
  Currently not used.

- MSK_IPAR_PRESOLVE_LINDEP_USE 304
  Controls whether the linear constraints are checked for linear dependencies.

- MSK_IPAR_PRESOLVE_LINDEP_WORK_LIM 305
  Controls linear dependency check in presolve.

- MSK_IPAR_PRESOLVE_USE 305
  Controls whether the presolve is applied to a problem before it is optimized.
- MSK_IPAR_READ_ADD_ANZ 305
  Controls how the constraint matrix is extended.
- MSK_IPAR_READ_ADD_CON 306
  Additional number of constraints that is made room for in the problem.
- MSK_IPAR_READ_ADD_CONE 306
  Additional number of conic constraints that is made room for in the problem.
- MSK_IPAR_READ_ADD_QNZ 306
  Controls how the quadratic matrixes are extended.
- MSK_IPAR_READ_ADD_VAR 306
  Additional number of variables that is made room for in the problem.
- MSK_IPAR_READ_ANZ 307
  Controls the expected number of constraint non-zeros.
- MSK_IPAR_READ_CON 307
  Controls the expected number of constraints.
- MSK_IPAR_READ_CONE 307
  Controls the expected number of conic constraints.
- MSK_IPAR_READ_DATA_COMPRESSED 308
  Controls the input file decompression.
- MSK_IPAR_READ_DATA_FORMAT 308
  Format of the data file to be read.
- MSK_IPAR_READ_KEEP_FREE_CON 308
  Controls whether the free constraints are included in the problem.
- MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU 309
  Controls how the LP files are interpreted.
- MSK_IPAR_READ_LP_QUOTED_NAMES 309
  If a name is in quotes when reading an LP file, the quotes will be removed.
- MSK_IPAR_READ_MPS_FORMAT 309
  Controls how strictly the MPS file reader interprets the MPS format.
- MSK_IPAR_READ_MPS_KEEP_INT 310
  Controls if integer constraints are read.

- `MSK_IPAR_READ_MPS_OBJ_SENSE` 310
  Controls the MPS format extensions.

- `MSK_IPAR_READ_MPS_QUOTED_NAMES` 310
  Controls the MPS format extensions.

- `MSK_IPAR_READ_MPS_RELAX` 311
  Controls the meaning of integer constraints.

- `MSK_IPAR_READ_MPS_WIDTH` 311
  Controls the maximal number of chars allowed in one line of the MPS file.

- `MSK_IPAR_READ_Q_MODE` 311
  Controls how the Q matrices are read from the MPS file.

- `MSK_IPAR_READ_QNZ` 312
  Controls the expected number of quadratic non-zeros.

- `MSK_IPAR_READ_TASK_IGNORE_PARAM` 312
  Controls what information is used from the task files.

- `MSK_IPAR_READ_VAR` 312
  Controls the expected number of variables.

- `MSK_IPAR_SENSITIVITY_OPTIMIZER` 313
  Controls which optimizer is used for optimal partition sensitivity analysis.

- `MSK_IPAR_SENSITIVITY_TYPE` 313
  Controls which type of sensitivity analysis is to be performed.

- `MSK_IPAR_SIM_DEGEN` 314
  Controls how aggressive degeneration is approached.

- `MSK_IPAR_SIM_DUAL_CRASH` 314
  Controls whether crashing is performed in the dual simplex optimizer.

- `MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION` 314
  Controls how aggressively restricted selection is used.

- `MSK_IPAR_SIM_DUAL_SELECTION` 315
  Controls the dual simplex strategy.

- `MSK_IPAR_SIM_HOTSTART` 315
  Controls the type of hot-start that the simplex optimizer perform.

- `MSK_IPAR_SIM_MAX_ITERATIONS` 316
  Maximum number of iterations that can be used by a simplex optimizer.

- MSK_IPAR_SIM_MAX_NUM_SETBACKS 316
  Controls how many setbacks that are allowed within a simplex optimizer.

- MSK_IPAR_SIM_NETWORK_DETECT 316
  Level of aggressiveness of network detection.

- MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART 317
  Level of aggressiveness of network detection in a simplex hot-start.

- MSK_IPAR_SIM_NETWORK_DETECT_METHOD 317
  Controls which type of detection method the network extraction should use.

- MSK_IPAR_SIM_NON_SINGULAR 317
  Controls if the simplex optimizer ensures a non-singular basis, if possible.

- MSK_IPAR_SIM_PRIMAL_CRASH 318
  Controls the simplex crash.

- MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION 318
  Controls how aggressively restricted selection is used.

- MSK_IPAR_SIM_PRIMAL_SELECTION 318
  Controls the primal simplex strategy.

- MSK_IPAR_SIM_REFACTOR_FREQ 319
  Controls the basis refactoring frequency.

- MSK_IPAR_SIM_SAVE_LU 319
  Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.

- MSK_IPAR_SIM_SCALING 320
  Controls how the problem is scaled before a simplex optimizer is used.

- MSK_IPAR_SIM_SOLVE_FORM 320
  Controls whether the primal or the dual problem is solved by the primal-/dual- simplex optimizer.

- MSK_IPAR_SIM_STABILITY_PRIORITY 320
  Controls how high priority the numerical stability should be given.

- MSK_IPAR_SIM_SWITCH_OPTIMIZER 321
  Controls the simplex behavior.

- MSK_IPAR_SOL_FILTER_KEEP_BASIC 321
  Controls the license manager client behavior.

- MSK_IPAR_SOL_FILTER_KEEP_RANGED 321
  Control the contents of the solution files.
- MSK_IPAR_SOL_QUOTED_NAMES 322
  Controls the solution file format.
- MSK_IPAR_SOL_READ_NAME_WIDTH 322
  Controls the input solution file format.
- MSK_IPAR_SOL_READ_WIDTH 322
  Controls the input solution file format.
- MSK_IPAR_SOLUTION_CALLBACK 323
  Indicates whether solution call-backs will be performed during the optimization.
- MSK_IPAR_WARNING_LEVEL 323
  Warning level.
- MSK_IPAR_WRITE_BAS_CONSTRAINTS 323
  Controls the basic solution file format.
- MSK_IPAR_WRITE_BAS_HEAD 324
  Controls the basic solution file format.
- MSK_IPAR_WRITE_BAS_VARIABLES 324
  Controls the basic solution file format.
- MSK_IPAR_WRITE_DATA_COMPRESSED 324
  Controls output file compression.
- MSK_IPAR_WRITE_DATA_FORMAT 325
  Controls the output file problem format.
- MSK_IPAR_WRITE_DATA_PARAM 325
  Controls output file data.
- MSK_IPAR_WRITE_FREE_CON 325
  Controls the output file data.
- MSK_IPAR_WRITE_GENERIC_NAMES 326
  Controls the output file data.
- MSK_IPAR_WRITE_GENERIC_NAMES_IO 326
  Index origin used in generic names.
- MSK_IPAR_WRITE_INT_CONSTRAINTS 326
  Controls the integer solution file format.

- MSK_IPAR_WRITE_INT_HEAD 327
  Controls the integer solution file format.

- MSK_IPAR_WRITE_INT_VARIABLES 327
  Controls the integer solution file format.

- MSK_IPAR_WRITE_LP_LINE_WIDTH 327
  Controls the LP output file format.

- MSK_IPAR_WRITE_LP_QUOTED_NAMES 327
  Controls LP output file format.

- MSK_IPAR_WRITE_LP_STRICT_FORMAT 328
  Controls whether LP output files satisfy the LP format strictly.

- MSK_IPAR_WRITE_LP_TERMS_PER_LINE 328
  Controls the LP output file format.

- MSK_IPAR_WRITE_MPS_INT 328
  Controls the output file data.

- MSK_IPAR_WRITE_MPS_OBJ_SENSE 329
  Controls the output file data.

- MSK_IPAR_WRITE_MPS_QUOTED_NAMES 329
  Controls the output file data.

- MSK_IPAR_WRITE_MPS_STRICT 329
  Controls the output MPS file format.

- MSK_IPAR_WRITE_PRECISION 330
  Controls data precision employed in when writing an MPS file.

- MSK_IPAR_WRITE_SOL_CONSTRAINTS 330
  Controls the solution file format.

- MSK_IPAR_WRITE_SOL_HEAD 330
  Controls solution file format.

- MSK_IPAR_WRITE_SOL_VARIABLES 331
  Controls the solution file format.

- MSK_IPAR_WRITE_TASK_INC_SOL 331
  Controls whether the solutions are stored in the task file too.

- `MSK_IPAR_WRITE_XML_MODE` 331
  Controls if linear coefficients should be written by row or column when writing in the XML file format.

- `alloc_add_qnz`

  Corresponding constant:
  `MSK_IPAR_ALLOC_ADD_QNZ`

  Description:
  Additional number of Q non-zeros that are allocated space for when `numanz` exceeds `maxnumqnz` during addition of new Q entries.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  5000

- `bi_clean_optimizer`

  Corresponding constant:
  `MSK_IPAR_BI_CLEAN_OPTIMIZER`

  Description:
  Controls which simplex optimizer is used in the clean-up phase.

  Possible Values:

  `MSK_OPTIMIZER_INTPNT` The interior-point optimizer is used.

  `MSK_OPTIMIZER_CONCURRENT` The optimizer for nonconvex nonlinear problems.

  `MSK_OPTIMIZER_MIXED_INT` The mixed integer optimizer.

  `MSK_OPTIMIZER_DUAL_SIMPLEX` The dual simplex optimizer is used.

  `MSK_OPTIMIZER_FREE` The optimizer is chosen automatically.

  `MSK_OPTIMIZER_CONIC` Another cone optimizer.

  `MSK_OPTIMIZER_NONCONVEX` The optimizer for nonconvex nonlinear problems.

  `MSK_OPTIMIZER_QCONE` The Qcone optimizer is used.

  `MSK_OPTIMIZER_PRIMAL_SIMPLEX` The primal simplex optimizer is used.

  `MSK_OPTIMIZER_FREE_SIMPLEX` Either the primal or the dual simplex optimizer is used.

  Default value:
  MSK_OPTIMIZER_FREE

- `bi_ignore_max_iter`

Corresponding constant:
`MSK_IPAR_BI_IGNORE_MAX_ITER`

Description:
If the parameter `MSK_IPAR_INTPNT_BASIS` has the value `MSK_BI_NO_ERROR` and the interior-point optimizer has terminated due to maximum number of iterations, then basis identification is performed if this parameter has the value `MSK_ON`.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `bi_ignore_num_error`

Corresponding constant:
`MSK_IPAR_BI_IGNORE_NUM_ERROR`

Description:
If the parameter `MSK_IPAR_INTPNT_BASIS` has the value `MSK_BI_NO_ERROR` and the interior-point optimizer has terminated due to a numerical problem, then basis identification is performed if this parameter has the value `MSK_ON`.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `bi_max_iterations`

Corresponding constant:
`MSK_IPAR_BI_MAX_ITERATIONS`

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

Possible Values:
Any number between 0 and +inf.

Default value:
1000000

- `cache_size_l1`

Corresponding constant:
`MSK_IPAR_CACHE_SIZE_L1`

Description:
Specifies the size of the cache of the computer. This parameter is potentially very important for the efficiency on computers if MOSEK cannot determine the cache size automatically. If the cache size is negative, then MOSEK tries to determine the value automatically.

Possible Values:
Any number between -inf and +inf.

Default value:
-1

- `cache_size_l2`

Corresponding constant:
`MSK_IPAR_CACHE_SIZE_L2`

Description:
Specifies the size of the cache of the computer. This parameter is potentially very important for the efficiency on computers where MOSEK cannot determine the cache size automatically. If the cache size is negative, then MOSEK tries to determine the value automatically.

Possible Values:
Any number between -inf and +inf.

Default value:
-1

- `check_convexity`

Corresponding constant:
`MSK_IPAR_CHECK_CONVEXITY`

Description:
Specify the level of convexity check on quadratic problems

Possible Values:

`MSK_CHECK_CONVEXITY_SIMPLE` Perform simple and fast convexity check.

`MSK_CHECK_CONVEXITY_NONE` No convexity check.

Default value:
MSK_CHECK_CONVEXITY_SIMPLE

- `check_ctrl_c`

Corresponding constant:
`MSK_IPAR_CHECK_CTRL_C`

Description:
Specifies whether MOSEK should check for `<ctrl>+<c>` key presses. In case it has, then control is returned to the user program.

In case a user-defined ctrl-c function is defined then that is used to check for ctrl-c. Otherwise the system procedure `signal` is used.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `check_task_data`

Corresponding constant:
`MSK_IPAR_CHECK_TASK_DATA`

Description:
If this feature is turned on, then the task data is checked for bad values i.e. NaNs. before an optimization is performed.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `concurrent_num_optimizers`

Corresponding constant:
`MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS`

Description:
The maximum number of simultaneous optimizations that will be started by the concurrent optimizer.

Possible Values:
Any number between 0 and +inf.

Default value:
2

- `concurrent_priority_dual_simplex`

Corresponding constant:
`MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX`

Description:
Priority of the dual simplex algorithm when selecting solvers for concurrent optimization.

Possible Values:
Any number between 0 and +inf.

Default value:
2

- `concurrent_priority_free_simplex`

  Corresponding constant:
  `MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX`

  Description:
  Priority of the free simplex optimizer when selecting solvers for concurrent optimization.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  3

- `concurrent_priority_intpnt`

  Corresponding constant:
  `MSK_IPAR_CONCURRENT_PRIORITY_INTPNT`

  Description:
  Priority of the interior-point algorithm when selecting solvers for concurrent optimization.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  4

- `concurrent_priority_primal_simplex`

  Corresponding constant:
  `MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX`

  Description:
  Priority of the primal simplex algorithm when selecting solvers for concurrent optimization.

Possible Values:
Any number between 0 and +inf.

Default value:
1

- `cpu_type`

  Corresponding constant:
  `MSK_IPAR_CPU_TYPE`

  Description:
  Specifies the CPU type. By default MOSEK tries to auto detect the CPU type. Therefore, we recommend to change this parameter only if the auto detection does not work properly.

  Possible Values:

  `MSK_CPU_POWERPC_G5` A G5 PowerPC CPU.

  `MSK_CPU_INTEL_PM` An Intel PM cpu.

  `MSK_CPU_GENERIC` An generic CPU type for the platform

  `MSK_CPU_UNKNOWN` An unknown CPU.

  `MSK_CPU_AMD_OPTERON` An AMD Opteron (64 bit).

  `MSK_CPU_INTEL_ITANIUM2` An Intel Itanium2.

  `MSK_CPU_AMD_ATHLON` An AMD Athlon.

  `MSK_CPU_HP_PARISC20` An HP PA RISC version 2.0 CPU.

  `MSK_CPU_INTEL_P4` An Intel Pentium P4 or Intel Xeon.

  `MSK_CPU_INTEL_P3` An Intel Pentium P3.

  `MSK_CPU_INTEL_CORE2` An Intel CORE2 cpu.

  Default value:
  MSK_CPU_UNKNOWN

- `data_check`

  Corresponding constant:
  `MSK_IPAR_DATA_CHECK`

  Description:
  If this option is turned on, then extensive data checking is enabled. It will slow down MOSEK but on the other hand help locating bugs.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `feasrepair_optimize`

  Corresponding constant:
  `MSK_IPAR_FEASREPAIR_OPTIMIZE`

  Description:
  Controls which type of feasibility analysis is to be performed.

  Possible Values:

  `MSK_FEASREPAIR_OPTIMIZE_NONE` Do not optimize the feasibility repair problem.

  `MSK_FEASREPAIR_OPTIMIZE_COMBINED` Minimize with original objective subject to minimal weighted violation of bounds.

  `MSK_FEASREPAIR_OPTIMIZE_PENALTY` Minimize weighted sum of violations.

  Default value:
  MSK_FEASREPAIR_OPTIMIZE_NONE

- `flush_stream_freq`

  Corresponding constant:
  `MSK_IPAR_FLUSH_STREAM_FREQ`

  Description:
  Controls how frequent the message and log streams are flushed. A value of 0 means that it is never flushed. Otherwise a larger value results in less frequent flushes.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  24

- `infeas_generic_names`

  Corresponding constant:
  `MSK_IPAR_INFEAS_GENERIC_NAMES`

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

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `infeas_prefer_primal`

  Corresponding constant:
  `MSK_IPAR_INFEAS_PREFER_PRIMAL`

  Description:
  If both certificates of primal and dual infeasibility are supplied then only the primal is used when this option is turned on.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `infeas_report_auto`

  Corresponding constant:
  `MSK_IPAR_INFEAS_REPORT_AUTO`

  Description:
  Controls whether an infeasibility report is automatically produced after the optimization if the problem is primal or dual infeasible.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `infeas_report_level`

  Corresponding constant:
  `MSK_IPAR_INFEAS_REPORT_LEVEL`

  Description:
  Controls the amount of information presented in an infeasibility report. Higher values imply more information.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  1

- `intpnt_basis`

Corresponding constant:

`MSK_IPAR_INTPNT_BASIS`

Description:

Controls whether the interior-point optimizer also computes an optimal basis.

Possible Values:

`MSK_BI_ALWAYS` Basis identification is always performed even if the interior-point optimizer terminates abnormally.

`MSK_BI_NO_ERROR` Basis identification is performed if the interior-point optimizer terminates without an error.

`MSK_BI_NEVER` Never do basis identification.

`MSK_BI_IF_FEASIBLE` Basis identification is not performed if the interior-point optimizer terminates with a problem status saying that the problem is primal or dual infeasible.

`MSK_BI_OTHER` Try another BI method.

Default value:

MSK_BI_ALWAYS

See also:

`MSK_IPAR_BI_IGNORE_MAX_ITER` Turns on basis identification in case the interior-point optimizer is terminated due to maximum number of iterations.

`MSK_IPAR_BI_IGNORE_NUM_ERROR` Turns on basis identification in case the interior-point optimizer is terminated due to a numerical problem.

- `intpnt_diff_step`

Corresponding constant:

`MSK_IPAR_INTPNT_DIFF_STEP`

Description:

Controls whether different step sizes are allowed in the primal and dual space.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

MSK_ON

- `intpnt_factor_debug_lvl`

Corresponding constant:

`MSK_IPAR_INTPNT_FACTOR_DEBUG_LVL`

Description:
Controls factorization debug level.

Possible Values:
Any number between 0 and +inf.

Default value:
0

- `intpnt_factor_method`

  Corresponding constant:
  `MSK_IPAR_INTPNT_FACTOR_METHOD`

  Description:
  Controls the method used to factor the Newton equation system.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `intpnt_max_iterations`

  Corresponding constant:
  `MSK_IPAR_INTPNT_MAX_ITERATIONS`

  Description:
  Controls the maximum number of iterations allowed in the interior-point optimizer.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  400

- `intpnt_max_num_cor`

  Corresponding constant:
  `MSK_IPAR_INTPNT_MAX_NUM_COR`

  Description:
  Controls the maximum number of correctors allowed by the multiple corrector procedure. A negative value means that MOSEK is making the choice.

  Possible Values:
  Any number between -1 and +inf.

  Default value:
  -1

- `intpnt_max_num_refinement_steps`

  Corresponding constant:
  `MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS`

  Description:
  Maximum number of steps to be used by the iterative refinement of the search direction. A negative value implies that the optimizer Chooses the maximum number of iterative refinement steps.

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1

- `intpnt_num_threads`

  Corresponding constant:
  `MSK_IPAR_INTPNT_NUM_THREADS`

  Description:
  Controls the number of threads employed by the interior-point optimizer.

  Possible Values:
  Any integer greater than 1.

  Default value:
  1

- `intpnt_off_col_trh`

  Corresponding constant:
  `MSK_IPAR_INTPNT_OFF_COL_TRH`

  Description:
  Controls how many offending columns are detected in the Jacobian of the constraint matrix.

  1 means aggressive detection, higher values mean less aggressive detection.

  0 means no detection.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  40

- `intpnt_order_method`

Corresponding constant:
`MSK_IPAR_INTPNT_ORDER_METHOD`

Description:
Controls the ordering strategy used by the interior-point optimizer when factorizing the Newton equation system.

Possible Values:

`MSK_ORDER_METHOD_NONE` No ordering is used.

`MSK_ORDER_METHOD_APPMINLOC2` A variant of the approximate minimum local-fill-in ordering is used.

`MSK_ORDER_METHOD_APPMINLOC1` Approximate minimum local-fill-in ordering is used.

`MSK_ORDER_METHOD_GRAPHPAR2` An alternative graph partitioning based ordering.

`MSK_ORDER_METHOD_FREE` The ordering method is chosen automatically.

`MSK_ORDER_METHOD_GRAPHPAR1` Graph partitioning based ordering.

Default value:
MSK_ORDER_METHOD_FREE

- `intpnt_regularization_use`

Corresponding constant:
`MSK_IPAR_INTPNT_REGULARIZATION_USE`

Description:
Controls whether regularization is allowed.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `intpnt_scaling`

Corresponding constant:
`MSK_IPAR_INTPNT_SCALING`

Description:
Controls how the problem is scaled before the interior-point optimizer is used.

Possible Values:

`MSK_SCALING_NONE` No scaling is performed.

`MSK_SCALING_MODERATE` A conservative scaling is performed.

`MSK_SCALING_AGGRESSIVE` A very aggressive scaling is performed.

`MSK_SCALING_FREE` The optimizer chooses the scaling heuristic.

Default value:
MSK_SCALING_FREE

- `intpnt_solve_form`

  Corresponding constant:
  `MSK_IPAR_INTPNT_SOLVE_FORM`

  Description:
  Controls whether the primal or the dual problem is solved.

  Possible Values:

  `MSK_SOLVE_PRIMAL` The optimizer should solve the primal problem.

  `MSK_SOLVE_DUAL` The optimizer should solve the dual problem.

  `MSK_SOLVE_FREE` The optimizer is free to solve either the primal or the dual problem.

  Default value:
  MSK_SOLVE_FREE

- `intpnt_starting_point`

  Corresponding constant:
  `MSK_IPAR_INTPNT_STARTING_POINT`

  Description:
  Starting point used by the interior-point optimizer.

  Possible Values:

  `MSK_STARTING_POINT_CONSTANT` The starting point is set to a constant. This is more reliable than a non-constant starting point.

  `MSK_STARTING_POINT_FREE` The starting point is chosen automatically.

  Default value:
  MSK_STARTING_POINT_FREE

- `license_allow_overuse`

  Corresponding constant:
  `MSK_IPAR_LICENSE_ALLOW_OVERUSE`

  Description:
  Controls if license overuse is allowed when caching licenses

  Possible Values:

  `MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `license_cache_time`

  Corresponding constant:
  `MSK_IPAR_LICENSE_CACHE_TIME`

  Description:
  Controls the amount of time a license is cached in the MOSEK environment for reuse. Checking out a license from the license server has a small overhead. Therefore, if a large number of optimizations is performed within a small amount of time, it is efficient to cache the license in the MOSEK environment for later use. This way a number of license check outs from the license server is avoided.

  If a license has not been used in the given amount of time,MOSEK will automatically check in the license. To disable license caching set the value to 0.

  Possible Values:
  Any number between 0 and 65555.

  Default value:
  5

- `license_check_time`

  Corresponding constant:
  `MSK_IPAR_LICENSE_CHECK_TIME`

  Description:
  The parameter specifies the number of seconds between the checks of all the active licenses in the MOSEK environment license cache. These checks are performed to determine if the licenses should be returned to the server.

  Possible Values:
  Any number between 1 and 120.

  Default value:
  1

- `license_debug`

  Corresponding constant:
  `MSK_IPAR_LICENSE_DEBUG`

  Description:
  This option is used to turn on debugging of the incense manager.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `license_pause_time`

Corresponding constant:
`MSK_IPAR_LICENSE_PAUSE_TIME`

Description:
If `MSK_IPAR_LICENSE_WAIT`=`MSK_ON` and no license is available, then MOSEK sleeps a number of micro seconds between each check of whether a license as become free.

Possible Values:
Any number between 0 and 1000000.

Default value:
100

- `license_suppress_expire_wrns`

Corresponding constant:
`MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS`

Description:
Controls whether license features expire warnings are suppressed.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `license_wait`

Corresponding constant:
`MSK_IPAR_LICENSE_WAIT`

Description:
If all licenses are in use MOSEK returns with an error code. However, by turning on this parameter MOSEK will wait for an available license.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

MSK_OFF

- `log`

Corresponding constant:

`MSK_IPAR_LOG`

Description:

Controls the amount of log information. The value 0 implies that all log information is suppressed. A higher level implies that more information is logged.

Please note that if a task is employed to solve a sequence of optimization problems the value of this parameter is reduced by the value of `MSK_IPAR_LOG_CUT_SECOND_OPT` for the second and any subsequent optimizations.

Possible Values:

Any number between 0 and +inf.

Default value:

10

See also:

`MSK_IPAR_LOG_CUT_SECOND_OPT` Controls the reduction in the log levels for the second and any subsequent optimizations.

- `log_bi`

Corresponding constant:

`MSK_IPAR_LOG_BI`

Description:

Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.

Possible Values:

Any number between 0 and +inf.

Default value:

4

- `log_bi_freq`

Corresponding constant:

`MSK_IPAR_LOG_BI_FREQ`

Description:

Controls how frequent the optimizer outputs information about the basis identification and how frequent the user-defined call-back function is called.

Possible Values:
Any number between 0 and +inf.

Default value:
2500

- `log_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

- `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.

- `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

- `log_feasrepair`

  Corresponding constant:
  `MSK_IPAR_LOG_FEASREPAIR`

  Description:
  Controls the amount of output printed when performing feasibility repair.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `log_file`

  Corresponding constant:
  `MSK_IPAR_LOG_FILE`

  Description:
  If turned on, then some log info is printed when a file is written or read.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  1

- `log_head`

  Corresponding constant:
  `MSK_IPAR_LOG_HEAD`

  Description:
  If turned on, then a header line is added to the log.

  Possible Values:
  Any number between 0 and +inf.

Default value:
1

- `log_infeas_ana`

  Corresponding constant:
  `MSK_IPAR_LOG_INFEAS_ANA`

  Description:
  Controls amount of output printed by the infeasibility analyzer procedures. A higher level implies that more information is logged.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  1

- `log_intpnt`

  Corresponding constant:
  `MSK_IPAR_LOG_INTPNT`

  Description:
  Controls amount of output printed printed by the interior-point optimizer. A higher level implies that more information is logged.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  4

- `log_mio`

  Corresponding constant:
  `MSK_IPAR_LOG_MIO`

  Description:
  Controls the log level for the mixed integer optimizer. A higher level implies that more information is logged.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  4

- `log_mio_freq`

Corresponding constant:
`MSK_IPAR_LOG_MIO_FREQ`

Description:
Controls how frequent the mixed integer optimizer prints the log line. It will print line every time `MSK_IPAR_LOG_MIO_FREQ` relaxations have been solved.

Possible Values:
A integer value.

Default value:
250

- `log_nonconvex`

Corresponding constant:
`MSK_IPAR_LOG_NONCONVEX`

Description:
Controls amount of output printed by the nonconvex optimizer.

Possible Values:
Any number between 0 and +inf.

Default value:
1

- `log_optimizer`

Corresponding constant:
`MSK_IPAR_LOG_OPTIMIZER`

Description:
Controls the amount of general optimizer information that is logged.

Possible Values:
Any number between 0 and +inf.

Default value:
1

- `log_order`

Corresponding constant:
`MSK_IPAR_LOG_ORDER`

Description:
If turned on, then factor lines are added to the log.

Possible Values:
Any number between 0 and +inf.

Default value:
1

- `log_param`

  Corresponding constant:
  `MSK_IPAR_LOG_PARAM`

  Description:
  Controls the amount of information printed out about parameter changes.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `log_presolve`

  Corresponding constant:
  `MSK_IPAR_LOG_PRESOLVE`

  Description:
  Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  1

- `log_response`

  Corresponding constant:
  `MSK_IPAR_LOG_RESPONSE`

  Description:
  Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `log_sensitivity`

  Corresponding constant:
  `MSK_IPAR_LOG_SENSITIVITY`

Description:
Controls the amount of logging during the sensitivity analysis. 0: Means no logging information is produced. 1: Timing information is printed. 2: Sensitivity results are printed.

Possible Values:
Any number between 0 and +inf.

Default value:
1

- `log_sensitivity_opt`

  Corresponding constant:
  `MSK_IPAR_LOG_SENSITIVITY_OPT`

  Description:
  Controls the amount of logging from the optimizers employed during the sensitivity analysis. 0 means no logging information is produced.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `log_sim`

  Corresponding constant:
  `MSK_IPAR_LOG_SIM`

  Description:
  Controls amount of output printed by the simplex optimizer. A higher level implies that more information is logged.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  4

- `log_sim_freq`

  Corresponding constant:
  `MSK_IPAR_LOG_SIM_FREQ`

  Description:
  Controls how frequent the simplex optimizer outputs information about the optimization and how frequent the user-defined call-back function is called.

Possible Values:
Any number between 0 and +inf.

Default value:
500

- `log_sim_minor`

  Corresponding constant:
  `MSK_IPAR_LOG_SIM_MINOR`

  Description:
  Currently not in use.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  1

- `log_sim_network_freq`

  Corresponding constant:
  `MSK_IPAR_LOG_SIM_NETWORK_FREQ`

  Description:
  Controls how frequent the network simplex optimizer outputs information about the optimization and how frequent the user-defined call-back function is called. The network optimizer will use a logging frequency equal to `MSK_IPAR_LOG_SIM_FREQ` times `MSK_IPAR_LOG_SIM_NETWORK_FREQ`.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  50

- `log_storage`

  Corresponding constant:
  `MSK_IPAR_LOG_STORAGE`

  Description:
  When turned on, MOSEK prints messages regarding the storage usage and allocation.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `lp_write_ignore_incompatible_items`

  Corresponding constant:
  `MSK_IPAR_LP_WRITE_IGNORE_INCOMPATIBLE_ITEMS`

  Description:
  Controls the result of writing a problem containing incompatible items to an LP file.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `max_num_warnings`

  Corresponding constant:
  `MSK_IPAR_MAX_NUM_WARNINGS`

  Description:
  Waning level. A higher value results in more warnings.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  10

- `maxnumanz_double_trh`

  Corresponding constant:
  `MSK_IPAR_MAXNUMANZ_DOUBLE_TRH`

  Description:
  Whenever MOSEK runs out of storage for the A matrix, it will double the value for `maxnumanz` until `maxnumnza` reaches the value of this parameter. When this threshold is reached it will use a slower increase.

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1

- `mio_branch_dir`

  Corresponding constant:
  `MSK_IPAR_MIO_BRANCH_DIR`

Description:
Controls whether the mixed integer optimizer is branching up or down by default.

Possible Values:

`MSK_BRANCH_DIR_DOWN` The mixed integer optimizer always chooses the down branch first.

`MSK_BRANCH_DIR_UP` The mixed integer optimizer always chooses the up branch first.

`MSK_BRANCH_DIR_FREE` The mixed optimizer decides which branch to choose.

Default value:
MSK_BRANCH_DIR_FREE

- `mio_branch_priorities_use`

Corresponding constant:
`MSK_IPAR_MIO_BRANCH_PRIORITIES_USE`

Description:
Controls whether branching priorities are used by the mixed integer optimizer.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `mio_construct_sol`

Corresponding constant:
`MSK_IPAR_MIO_CONSTRUCT_SOL`

Description:
If set to `MSK_ON` and all integer variables have been given a value for which a feasible MIP solution exists, then MOSEK generates an initial solution to the MIP by fixing all integer values and solving for the continuous variables.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `mio_cont_sol`

Corresponding constant:

`MSK_IPAR_MIO_CONT_SOL`

Description:

Controls the meaning of the interior-point and basic solutions in MIP problems.

Possible Values:

`MSK_MIO_CONT_SOL_ITG` The reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. A solution is only reported in case the problem has a primal feasible solution.

`MSK_MIO_CONT_SOL_NONE` No interior-point or basic solution are reported when the mixed integer optimizer is used.

`MSK_MIO_CONT_SOL_ROOT` The reported interior-point and basic solutions are a solution to the root node problem when mixed integer optimizer is used.

`MSK_MIO_CONT_SOL_ITG_REL` In case the problem is primal feasible then the reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. If the problem is primal infeasible, then the solution to the root node problem is reported.

Default value:

MSK_MIO_CONT_SOL_NONE

- `mio_cut_level_root`

Corresponding constant:

`MSK_IPAR_MIO_CUT_LEVEL_ROOT`

Description:

Controls the cut level employed by the mixed integer optimizer at the root node. A negative value means a default value determined by the mixed integer optimizer is used. By adding the appropriate values from the following table the employed cut types can be controlled.

GUB cover	+2
Flow cover	+4
Lifting	+8
Plant location	+16
Disaggregation	+32
Knapsack cover	+64
Lattice	+128
Gomory	+256
Coefficient reduction	+512
GCD	+1024
Obj. integrality	+2048

Possible Values:

Any value.

Default value:

-1

- `mio_cut_level_tree`

  Corresponding constant:

  `MSK_IPAR_MIO_CUT_LEVEL_TREE`

  Description:

  Controls the cut level employed by the mixed integer optimizer at the tree. See `MSK_IPAR_MIO_CUT_LEVEL_ROOT` for an explanation of the parameter values.

  Possible Values:

  Any value.

  Default value:

  -1

- `mio_feaspump_level`

  Corresponding constant:

  `MSK_IPAR_MIO_FEASPUMP_LEVEL`

  Description:

  Feasibility pump is a heuristic designed to compute an initial feasible solution. A value of 0 implies that the feasibility pump heuristic is not used. A value of -1 implies that the mixed integer optimizer decides how the feasibility pump heuristic is used. A larger value than 1 implies that the feasibility pump is employed more aggressively. Normally a value beyond 3 is not worthwhile.

  Possible Values:

  Any number between -inf and 3.

  Default value:

  -1

- `mio_heuristic_level`

  Corresponding constant:

  `MSK_IPAR_MIO_HEURISTIC_LEVEL`

  Description:

  Controls the heuristic employed by the mixed integer optimizer to locate an initial good integer feasible solution. A value of zero means the heuristic is not used at all. A larger value than 0 means that a gradually more sophisticated heuristic is used which is computationally more expensive. A negative value implies that the optimizer chooses the heuristic. Normally a value around 3 to 5 should be optimal.

Possible Values:
Any value.

Default value:
-1

- `mio_keep_basis`

  Corresponding constant:
  `MSK_IPAR_MIO_KEEP_BASIS`

  Description:
  Controls whether the integer presolve keeps bases in memory. This speeds on the solution process at cost of bigger memory consumption.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `mio_local_branch_number`

  Corresponding constant:
  `MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER`

  Description:

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1

- `mio_max_num_branches`

  Corresponding constant:
  `MSK_IPAR_MIO_MAX_NUM_BRANCHES`

  Description:
  Maximum number of branches allowed during the branch and bound search. A negative value means infinite.

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1

See also:

`MSK_DPAR_MIO_DISABLE_TERM_TIME` Certain termination criterias is disabled within the mixed integer optimizer for period time specified by the parameter.

- `mio_max_num_relaxs`

  Corresponding constant:
  `MSK_IPAR_MIO_MAX_NUM_RELAXS`

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

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1

  See also:

  `MSK_DPAR_MIO_DISABLE_TERM_TIME` Certain termination criterias is disabled within the mixed integer optimizer for period time specified by the parameter.

- `mio_max_num_solutions`

  Corresponding constant:
  `MSK_IPAR_MIO_MAX_NUM_SOLUTIONS`

  Description:
  The mixed integer optimizer can be terminated after a certain number of different feasible solutions has been located. If this parameter has the value n and n is strictly positive, then the mixed integer optimizer will be terminated when n feasible solutions have been located.

  Possible Values:
  Any number between -inf and +inf.

  Default value:
  -1

  See also:

  `MSK_DPAR_MIO_DISABLE_TERM_TIME` Certain termination criterias is disabled within the mixed integer optimizer for period time specified by the parameter.

- `mio_mode`

  Corresponding constant:
  `MSK_IPAR_MIO_MODE`

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

Possible Values:

`MSK_MIO_MODE_IGNORED` The integer constraints are ignored and the problem is solved as a continuous problem.

`MSK_MIO_MODE_LAZY` Integer restrictions should be satisfied if an optimizer is available for the problem.

`MSK_MIO_MODE_SATISFIED` Integer restrictions should be satisfied.

Default value:
MSK_MIO_MODE_SATISFIED

- `mio_node_optimizer`

Corresponding constant:
`MSK_IPAR_MIO_NODE_OPTIMIZER`

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

Possible Values:

`MSK_OPTIMIZER_INTPNT` The interior-point optimizer is used.

`MSK_OPTIMIZER_CONCURRENT` The optimizer for nonconvex nonlinear problems.

`MSK_OPTIMIZER_MIXED_INT` The mixed integer optimizer.

`MSK_OPTIMIZER_DUAL_SIMPLEX` The dual simplex optimizer is used.

`MSK_OPTIMIZER_FREE` The optimizer is chosen automatically.

`MSK_OPTIMIZER_CONIC` Another cone optimizer.

`MSK_OPTIMIZER_NONCONVEX` The optimizer for nonconvex nonlinear problems.

`MSK_OPTIMIZER_QCONE` The Qcone optimizer is used.

`MSK_OPTIMIZER_PRIMAL_SIMPLEX` The primal simplex optimizer is used.

`MSK_OPTIMIZER_FREE_SIMPLEX` Either the primal or the dual simplex optimizer is used.

Default value:
MSK_OPTIMIZER_FREE

- `mio_node_selection`

Corresponding constant:
`MSK_IPAR_MIO_NODE_SELECTION`

Description:
Controls the node selection strategy employed by the mixed integer optimizer.

Possible Values:

`MSK_MIO_NODE_SELECTION_PSEUDO` The optimizer employs selects the node based on a pseudo cost estimate.

`MSK_MIO_NODE_SELECTION_HYBRID` The optimizer employs a hybrid strategy.

`MSK_MIO_NODE_SELECTION_FREE` The optimizer decides the node selection strategy.

`MSK_MIO_NODE_SELECTION_WORST` The optimizer employs a worst bound node selection strategy.

`MSK_MIO_NODE_SELECTION_BEST` The optimizer employs a best bound node selection strategy.

`MSK_MIO_NODE_SELECTION_FIRST` The optimizer employs a depth first node selection strategy.

Default value:
MSK_MIO_NODE_SELECTION_FREE

- `mio_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_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `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_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `mio_presolve_use`

  Corresponding constant:
  `MSK_IPAR_MIO_PRESOLVE_USE`

  Description:
  Controls whether presolve is performed by the mixed integer optimizer.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `mio_root_optimizer`

  Corresponding constant:
  `MSK_IPAR_MIO_ROOT_OPTIMIZER`

  Description:
  Controls which optimizer is employed at the root node in the mixed integer optimizer.

  Possible Values:

  `MSK_OPTIMIZER_INTPNT` The interior-point optimizer is used.

  `MSK_OPTIMIZER_CONCURRENT` The optimizer for nonconvex nonlinear problems.

  `MSK_OPTIMIZER_MIXED_INT` The mixed integer optimizer.

  `MSK_OPTIMIZER_DUAL_SIMPLEX` The dual simplex optimizer is used.

  `MSK_OPTIMIZER_FREE` The optimizer is chosen automatically.

  `MSK_OPTIMIZER_CONIC` Another cone optimizer.

  `MSK_OPTIMIZER_NONCONVEX` The optimizer for nonconvex nonlinear problems.

  `MSK_OPTIMIZER_QCONE` The Qcone optimizer is used.

  `MSK_OPTIMIZER_PRIMAL_SIMPLEX` The primal simplex optimizer is used.

  `MSK_OPTIMIZER_FREE_SIMPLEX` Either the primal or the dual simplex optimizer is used.

  Default value:
  MSK_OPTIMIZER_FREE

- `mio_strong_branch`

Corresponding constant:
`MSK_IPAR_MIO_STRONG_BRANCH`

Description:
The value specifies the depth from the root in which strong branching is used. A negative value means that the optimizer chooses a default value automatically.

Possible Values:
Any number between -inf and +inf.

Default value:
-1

- `nonconvex_max_iterations`

Corresponding constant:
`MSK_IPAR_NONCONVEX_MAX_ITERATIONS`

Description:
Maximum number of iterations that can be used by the nonconvex optimizer.

Possible Values:
Any number between 0 and +inf.

Default value:
100000

- `objective_sense`

Corresponding constant:
`MSK_IPAR_OBJECTIVE_SENSE`

Description:
If the objective sense for the task is undefined, then the value of this parameter is used as the default objective sense.

Possible Values:

`MSK_OBJECTIVE_SENSE_MINIMIZE`	The problem should be minimized.
`MSK_OBJECTIVE_SENSE_UNDEFINED`	The objective sense is undefined.
`MSK_OBJECTIVE_SENSE_MAXIMIZE`	The problem should be maximized.

Default value:
MSK_OBJECTIVE_SENSE_MINIMIZE

- `opf_max_terms_per_line`

Corresponding constant:
`MSK_IPAR_OPF_MAX_TERMS_PER_LINE`

Description:
The maximum number of terms (linear and quadratic) per line when an OPF file is written.

Possible Values:
Any number between 0 and +inf.

Default value:
5

- `opf_write_header`

  Corresponding constant:
  `MSK_IPAR_OPF_WRITE_HEADER`

  Description:
  Write a text header with date and MOSEK version in an OPF file.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `opf_write_hints`

  Corresponding constant:
  `MSK_IPAR_OPF_WRITE_HINTS`

  Description:
  Write a hint section with problem dimensions in the beginning of an OPF file.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `opf_write_parameters`

  Corresponding constant:
  `MSK_IPAR_OPF_WRITE_PARAMETERS`

  Description:
  Write a parameter section in an OPF file.

  Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

MSK_OFF

- `opf_write_problem`

Corresponding constant:

`MSK_IPAR_OPF_WRITE_PROBLEM`

Description:

Write objective, constraints, bounds etc. to an OPF file.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

MSK_ON

- `opf_write_sol_bas`

Corresponding constant:

`MSK_IPAR_OPF_WRITE_SOL_BAS`

Description:

If `MSK_IPAR_OPF_WRITE_SOLUTIONS` is `MSK_ON` and a basic solution is defined, include the basic solution in OPF files.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

MSK_ON

- `opf_write_sol_itg`

Corresponding constant:

`MSK_IPAR_OPF_WRITE_SOL_ITG`

Description:

If `MSK_IPAR_OPF_WRITE_SOLUTIONS` is `MSK_ON` and an integer solution is defined, write the integer solution in OPF files.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `opf_write_sol_itr`

  Corresponding constant:
  `MSK_IPAR_OPF_WRITE_SOL_ITR`

  Description:
  If `MSK_IPAR_OPF_WRITE_SOLUTIONS` is `MSK_ON` and an interior solution is defined, write the interior solution in OPF files.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `opf_write_solutions`

  Corresponding constant:
  `MSK_IPAR_OPF_WRITE_SOLUTIONS`

  Description:
  Enable inclusion of solutions in the OPF files.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `optimizer`

  Corresponding constant:
  `MSK_IPAR_OPTIMIZER`

  Description:
  Controls which optimizer is used to optimize the task.

  Possible Values:

  `MSK_OPTIMIZER_INTPNT` The interior-point optimizer is used.

  `MSK_OPTIMIZER_CONCURRENT` The optimizer for nonconvex nonlinear problems.

  `MSK_OPTIMIZER_MIXED_INT` The mixed integer optimizer.

`MSK_OPTIMIZER_DUAL_SIMPLEX` The dual simplex optimizer is used.

`MSK_OPTIMIZER_FREE` The optimizer is chosen automatically.

`MSK_OPTIMIZER_CONIC` Another cone optimizer.

`MSK_OPTIMIZER_NONCONVEX` The optimizer for nonconvex nonlinear problems.

`MSK_OPTIMIZER_QCONE` The Qcone optimizer is used.

`MSK_OPTIMIZER_PRIMAL_SIMPLEX` The primal simplex optimizer is used.

`MSK_OPTIMIZER_FREE_SIMPLEX` Either the primal or the dual simplex optimizer is used.

Default value:

MSK_OPTIMIZER_FREE

- `param_read_case_name`

Corresponding constant:

`MSK_IPAR_PARAM_READ_CASE_NAME`

Description:

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

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

MSK_ON

- `param_read_ign_error`

Corresponding constant:

`MSK_IPAR_PARAM_READ_IGN_ERROR`

Description:

If turned on, then errors in paramter settings is ignored.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

MSK_OFF

- `presolve_elim_fill`

Corresponding constant:

`MSK_IPAR_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

- `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_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `presolve_level`

Corresponding constant:
`MSK_IPAR_PRESOLVE_LEVEL`

Description:
Currently not used.

Possible Values:
Any number between -inf and +inf.

Default value:
-1

- `presolve_lindep_use`

Corresponding constant:
`MSK_IPAR_PRESOLVE_LINDEP_USE`

Description:
Controls whether the linear constraints are checked for linear dependencies.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

MSK_ON

- `presolve_lindep_work_lim`

  Corresponding constant:

  `MSK_IPAR_PRESOLVE_LINDEP_WORK_LIM`

  Description:

  Is used to limit the amount of work that can done to locate linear dependencies. In general the higher value this parameter is given the less work can be used. However, a value of 0 means no limit on the amount work that can be used.

  Possible Values:

  Any number between 0 and +inf.

  Default value:

  1

- `presolve_use`

  Corresponding constant:

  `MSK_IPAR_PRESOLVE_USE`

  Description:

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

  Possible Values:

  `MSK_PRESOLVE_MODE_ON` The problem is presolved before it is optimized.

  `MSK_PRESOLVE_MODE_OFF` The problem is not presolved before it is optimized.

  `MSK_PRESOLVE_MODE_FREE` It is decided automatically whether to presolve before the problem is optimized.

  Default value:

  MSK_PRESOLVE_MODE_FREE

- `read_add_anz`

  Corresponding constant:

  `MSK_IPAR_READ_ADD_ANZ`

  Description:

  Additional number of non-zeros in A that is made room for in the problem.

  Possible Values:

  Any number between 0 and +inf.

Default value:
0

- `read_add_con`

  Corresponding constant:
  `MSK_IPAR_READ_ADD_CON`

  Description:
  Additional number of constraints that is made room for in the problem.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `read_add_cone`

  Corresponding constant:
  `MSK_IPAR_READ_ADD_CONE`

  Description:
  Additional number of conic constraints that is made room for in the problem.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `read_add_qnz`

  Corresponding constant:
  `MSK_IPAR_READ_ADD_QNZ`

  Description:
  Additional number of non-zeros in the Q matrices that is made room for in the problem.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  0

- `read_add_var`

  Corresponding constant:
  `MSK_IPAR_READ_ADD_VAR`

Description:
Additional number of variables that is made room for in the problem.

Possible Values:
Any number between 0 and +inf.

Default value:
0

- `read_anz`

  Corresponding constant:
  `MSK_IPAR_READ_ANZ`

  Description:
  Expected maximum number of A non-zeros to be read. The option is used only by fast MPS and LP file readers.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  100000

- `read_con`

  Corresponding constant:
  `MSK_IPAR_READ_CON`

  Description:
  Expected maximum number of constraints to be read. The option is only used by fast MPS and LP file readers.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  10000

- `read_cone`

  Corresponding constant:
  `MSK_IPAR_READ_CONE`

  Description:
  Expected maximum number of conic constraints to be read. The option is used only by fast MPS and LP file readers.

  Possible Values:
  Any number between 0 and +inf.

Default value:
2500

- `read_data_compressed`

  Corresponding constant:
  `MSK_IPAR_READ_DATA_COMPRESSED`

  Description:
  If this option is turned on,it is assumed that the data file is compressed.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `read_data_format`

  Corresponding constant:
  `MSK_IPAR_READ_DATA_FORMAT`

  Description:
  Format of the data file to be read.

  Possible Values:

  `MSK_DATA_FORMAT_XML` The data file is an XML formatted file.

  `MSK_DATA_FORMAT_EXTENSION` The file extension is used to determine the data file format.

  `MSK_DATA_FORMAT_MPS` The data file is MPS formatted.

  `MSK_DATA_FORMAT_LP` The data file is LP formatted.

  `MSK_DATA_FORMAT_MBT` The data file is a MOSEK binary task file.

  `MSK_DATA_FORMAT_OP` The data file is an optimization problem formatted file.

  Default value:
  MSK_DATA_FORMAT_EXTENSION

- `read_keep_free_con`

  Corresponding constant:
  `MSK_IPAR_READ_KEEP_FREE_CON`

  Description:
  Controls whether the free constraints are included in the problem.

  Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `read_lp_drop_new_vars_in_bou`

  Corresponding constant:
  `MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU`

  Description:
  If this option is turned on, MOSEK will drop variables that are defined for the first time in the bounds section.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `read_lp_quoted_names`

  Corresponding constant:
  `MSK_IPAR_READ_LP_QUOTED_NAMES`

  Description:
  If a name is in quotes when reading an LP file, the quotes will be removed.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `read_mps_format`

  Corresponding constant:
  `MSK_IPAR_READ_MPS_FORMAT`

  Description:
  Controls how strictly the MPS file reader interprets the MPS format.

  Possible Values:

  `MSK_MPS_FORMAT_STRICT` It is assumed that the input file satisfies the MPS format strictly.

`MSK_MPS_FORMAT_RELAXED` It is assumed that the input file satisfies a slightly relaxed version of the MPS format.

`MSK_MPS_FORMAT_FREE` It is assumed that the input file satisfies the free MPS format. This implies that spaces are not allowed in names. Otherwise the format is free.

Default value:

MSK_MPS_FORMAT_RELAXED

- `read_mps_keep_int`

Corresponding constant:

`MSK_IPAR_READ_MPS_KEEP_INT`

Description:

Controls whether MOSEK should keep the integer restrictions on the variables while reading the MPS file.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

MSK_ON

- `read_mps_obj_sense`

Corresponding constant:

`MSK_IPAR_READ_MPS_OBJ_SENSE`

Description:

If turned on, the MPS reader uses the objective sense section. Otherwise the MPS reader ignores it.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

MSK_ON

- `read_mps_quoted_names`

Corresponding constant:

`MSK_IPAR_READ_MPS_QUOTED_NAMES`

Description:

If a name is in quotes when reading an MPS file, then the quotes will be removed.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `read_mps_relax`

  Corresponding constant:
  `MSK_IPAR_READ_MPS_RELAX`

  Description:
  If this option is turned on, then the relaxation of the MIP will be read.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `read_mps_width`

  Corresponding constant:
  `MSK_IPAR_READ_MPS_WIDTH`

  Description:
  Controls the maximal number of chars allowed in one line of the MPS file.

  Possible Values:
  Any positive number greater than 80.

  Default value:
  1024

- `read_q_mode`

  Corresponding constant:
  `MSK_IPAR_READ_Q_MODE`

  Description:
  Controls how the Q matrices are read from the MPS file.

  Possible Values:

  `MSK_Q_READ_ADD` All elements in a Q matrix are assumed to belong to the lower triangular part. Duplicate elements in a Q matrix are added together.

`MSK_Q_READ_DROP_LOWER` All elements in the strict lower triangular part of the Q matrices are dropped.

`MSK_Q_READ_DROP_UPPER` All elements in the strict upper triangular part of the Q matrices are dropped.

Default value:
MSK_Q_READ_ADD

- `read_qnz`

  Corresponding constant:
  `MSK_IPAR_READ_QNZ`

  Description:
  Expected maximum number of Q non-zeros to be read. The option is used only by MPS and LP file readers.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  20000

- `read_task_ignore_param`

  Corresponding constant:
  `MSK_IPAR_READ_TASK_IGNORE_PARAM`

  Description:
  Controls whether MOSEK should ignore the parameter setting defined in the task file and use the default parameter setting instead.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `read_var`

  Corresponding constant:
  `MSK_IPAR_READ_VAR`

  Description:
  Expected maximum number of variable to be read. The option is used only by MPS and LP file readers.

Possible Values:
Any number between 0 and +inf.

Default value:
10000

- `sensitivity_optimizer`

  Corresponding constant:
  `MSK_IPAR_SENSITIVITY_OPTIMIZER`

  Description:
  Controls which optimizer is used for optimal partition sensitivity analysis.

  Possible Values:

  `MSK_OPTIMIZER_INTPNT` The interior-point optimizer is used.

  `MSK_OPTIMIZER_CONCURRENT` The optimizer for nonconvex nonlinear problems.

  `MSK_OPTIMIZER_MIXED_INT` The mixed integer optimizer.

  `MSK_OPTIMIZER_DUAL_SIMPLEX` The dual simplex optimizer is used.

  `MSK_OPTIMIZER_FREE` The optimizer is chosen automatically.

  `MSK_OPTIMIZER_CONIC` Another cone optimizer.

  `MSK_OPTIMIZER_NONCONVEX` The optimizer for nonconvex nonlinear problems.

  `MSK_OPTIMIZER_QCONE` The Qcone optimizer is used.

  `MSK_OPTIMIZER_PRIMAL_SIMPLEX` The primal simplex optimizer is used.

  `MSK_OPTIMIZER_FREE_SIMPLEX` Either the primal or the dual simplex optimizer is used.

  Default value:
  MSK_OPTIMIZER_FREE_SIMPLEX

- `sensitivity_type`

  Corresponding constant:
  `MSK_IPAR_SENSITIVITY_TYPE`

  Description:
  Controls which type of sensitivity analysis is to be performed.

  Possible Values:

  `MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION` Optimal partition sensitivity analysis is performed.

  `MSK_SENSITIVITY_TYPE_BASIS` Basis sensitivity analysis is performed.

  Default value:
  MSK_SENSITIVITY_TYPE_BASIS

- `sim_degen`

  Corresponding constant:
  `MSK_IPAR_SIM_DEGEN`

  Description:
  Controls how aggressive degeneration is approached.

  Possible Values:

  `MSK_SIM_DEGEN_NONE` The simplex optimizer should use no degeneration strategy.

  `MSK_SIM_DEGEN_MODERATE` The simplex optimizer should use a moderate degeneration strategy.

  `MSK_SIM_DEGEN_MINIMUM` The simplex optimizer should use a minimum degeneration strategy.

  `MSK_SIM_DEGEN_AGGRESSIVE` The simplex optimizer should use an aggressive degeneration strategy.

  `MSK_SIM_DEGEN_FREE` The simplex optimizer chooses the degeneration strategy.

  Default value:
  MSK_SIM_DEGEN_FREE

- `sim_dual_crash`

  Corresponding constant:
  `MSK_IPAR_SIM_DUAL_CRASH`

  Description:
  Controls whether crashing is performed in the dual simplex optimizer.

  In general if a basis consists of more than (100-this parameter value)% fixed variables, then a crash will be performed.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  90

- `sim_dual_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

- `sim_dual_selection`

Corresponding constant:

`MSK_IPAR_SIM_DUAL_SELECTION`

Description:

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

Possible Values:

`MSK_SIM_SELECTION_FULL` The optimizer uses full pricing.

`MSK_SIM_SELECTION_PARTIAL` The optimizer uses a partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.

`MSK_SIM_SELECTION_FREE` The optimizer chooses the pricing strategy.

`MSK_SIM_SELECTION_ASE` The optimizer uses approximate steepest-edge pricing.

`MSK_SIM_SELECTION_DEVEX` The optimizer uses devex steepest-edge pricing (or if it is not available an approximate steep-edge selection).

`MSK_SIM_SELECTION_SE` The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

Default value:

MSK_SIM_SELECTION_FREE

- `sim_hotstart`

Corresponding constant:

`MSK_IPAR_SIM_HOTSTART`

Description:

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

Possible Values:

`MSK_SIM_HOTSTART_NONE` The simplex optimizer performs a coldstart.

`MSK_SIM_HOTSTART_STATUS_KEYS` Only the status keys of the constraints and variables are used to choose the type of hot-start.

`MSK_SIM_HOTSTART_FREE` The simplex optimize chooses the hot-start type.

Default value:

MSK_SIM_HOTSTART_FREE

- `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

- `sim_max_num_setbacks`

  Corresponding constant:

  `MSK_IPAR_SIM_MAX_NUM_SETBACKS`

  Description:

  Controls how many setbacks are allowed within a simplex optimizer. A setback 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

- `sim_network_detect`

  Corresponding constant:

  `MSK_IPAR_SIM_NETWORK_DETECT`

  Description:

  The simplex optimizer is capable of exploiting a network flow component in a problem. However it is only worthwhile to exploit the network flow component if it is sufficiently large. This parameter controls how large the network component has to be in “relative” terms before it is exploited. For instance a value of 20 means at least 20% of the model should be a network before it is exploited. If this value is larger than 100 the network flow component is never detected or exploited.

Possible Values:
Any number between 0 and +inf.

Default value:
101

- `sim_network_detect_hotstart`

  Corresponding constant:
  `MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART`

  Description:
  This parameter controls has large the network component in "relative" terms has to be before it is exploited in a simplex hot-start. The network component should be equal or larger than

  `max(MSK_IPAR_SIM_NETWORK_DETECT,MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART)`

  before it is exploited. If this value is larger than 100 the network flow component is never detected or exploited.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  100

- `sim_network_detect_method`

  Corresponding constant:
  `MSK_IPAR_SIM_NETWORK_DETECT_METHOD`

  Description:
  Controls which type of detection method the network extraction should use.

  Possible Values:

  `MSK_NETWORK_DETECT_SIMPLE` The network detection should use a very simple heuristic.

  `MSK_NETWORK_DETECT_ADVANCED` The network detection should use a more advanced heuristic.

  `MSK_NETWORK_DETECT_FREE` The network detection is free.

  Default value:
  MSK_NETWORK_DETECT_FREE

- `sim_non_singular`

  Corresponding constant:
  `MSK_IPAR_SIM_NON_SINGULAR`

Description:

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

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

MSK_ON

- `sim_primal_crash`

  Corresponding constant:

  `MSK_IPAR_SIM_PRIMAL_CRASH`

  Description:

  Controls whether crashing is performed in the primal simplex optimizer.

  In general, if a basis consists of more than (100-this parameter value)% fixed variables, then a crash will be performed.

  Possible Values:

  Any nonnegative integer value.

  Default value:

  90

- `sim_primal_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

- `sim_primal_selection`

Corresponding constant:

`MSK_IPAR_SIM_PRIMAL_SELECTION`

Description:

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

Possible Values:

`MSK_SIM_SELECTION_FULL` The optimizer uses full pricing.

`MSK_SIM_SELECTION_PARTIAL` The optimizer uses a partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.

`MSK_SIM_SELECTION_FREE` The optimizer chooses the pricing strategy.

`MSK_SIM_SELECTION_ASE` The optimizer uses approximate steepest-edge pricing.

`MSK_SIM_SELECTION_DEVEX` The optimizer uses devex steepest-edge pricing (or if it is not available an approximate steep-edge selection).

`MSK_SIM_SELECTION_SE` The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

Default value:

MSK_SIM_SELECTION_FREE

- `sim_refactor_freq`

Corresponding constant:

`MSK_IPAR_SIM_REFACTOR_FREQ`

Description:

Controls how frequent the basis is refactorized. The value 0 means that the optimizer determines the best point of refactorization.

It is strongly recommended NOT to change this parameter.

Possible Values:

Any number between 0 and +inf.

Default value:

0

- `sim_save_lu`

Corresponding constant:

`MSK_IPAR_SIM_SAVE_LU`

Description:

Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

MSK_OFF

- `sim_scaling`

Corresponding constant:

`MSK_IPAR_SIM_SCALING`

Description:

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

Possible Values:

`MSK_SCALING_NONE` No scaling is performed.

`MSK_SCALING_MODERATE` A conservative scaling is performed.

`MSK_SCALING_AGGRESSIVE` A very aggressive scaling is performed.

`MSK_SCALING_FREE` The optimizer chooses the scaling heuristic.

Default value:

MSK_SCALING_FREE

- `sim_solve_form`

Corresponding constant:

`MSK_IPAR_SIM_SOLVE_FORM`

Description:

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

Possible Values:

`MSK_SOLVE_PRIMAL` The optimizer should solve the primal problem.

`MSK_SOLVE_DUAL` The optimizer should solve the dual problem.

`MSK_SOLVE_FREE` The optimizer is free to solve either the primal or the dual problem.

Default value:

MSK_SOLVE_FREE

- `sim_stability_priority`

Corresponding constant:

`MSK_IPAR_SIM_STABILITY_PRIORITY`

Description:
Controls how high priority the numerical stability should be given.

Possible Values:
Any number between 0 and 100.

Default value:
50

- `sim_switch_optimizer`

  Corresponding constant:
  `MSK_IPAR_SIM_SWITCH_OPTIMIZER`

  Description:
  The simplex optimizer sometimes chooses to solve the dual problem instead of the primal problem. This implies that if you have chosen to use the dual simplex optimizer and the problem is dualized, then it actually makes sense to use the primal simplex optimizer instead. If this parameter is on and the problem is dualized and furthermore the simplex optimizer is chosen to be the primal (dual) one, then it is switched to the dual (primal).

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `sol_filter_keep_basic`

  Corresponding constant:
  `MSK_IPAR_SOL_FILTER_KEEP_BASIC`

  Description:
  If turned on, then basic and super basic constraints and variables are written to the solution file independent of the filter setting.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `sol_filter_keep_ranged`

Corresponding constant:
`MSK_IPAR_SOL_FILTER_KEEP_RANGED`

Description:
If turned on, then ranged constraints and variables are written to the solution file independent of the filter setting.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `sol_quoted_names`

Corresponding constant:
`MSK_IPAR_SOL_QUOTED_NAMES`

Description:
If this options is turned on, then MOSEK will quote names that contains blanks while writing the solution file. Moreover when reading leading and trailing quotes will be stripped of.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `sol_read_name_width`

Corresponding constant:
`MSK_IPAR_SOL_READ_NAME_WIDTH`

Description:
When a solution is read by MOSEK and some constraint, variable or cone names contain blanks, then a maximum name width much be specified. A negative value implies that no name contain blanks.

Possible Values:
Any number between -inf and +inf.

Default value:
-1

- `sol_read_width`

Corresponding constant:
`MSK_IPAR_SOL_READ_WIDTH`

Description:
Controls the maximal acceptable width of line in the solutions when read by MOSEK.

Possible Values:
Any positive number greater than 80.

Default value:
1024

- `solution_callback`

  Corresponding constant:
  `MSK_IPAR_SOLUTION_CALLBACK`

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

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `warning_level`

  Corresponding constant:
  `MSK_IPAR_WARNING_LEVEL`

  Description:
  Warning level.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  1

- `write_bas_constraints`

  Corresponding constant:
  `MSK_IPAR_WRITE_BAS_CONSTRAINTS`

  Description:
  Controls whether the constraint section is written to the basic solution file.

  Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `write_bas_head`

  Corresponding constant:
  `MSK_IPAR_WRITE_BAS_HEAD`

  Description:
  Controls whether the header section is written to the basic solution file.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `write_bas_variables`

  Corresponding constant:
  `MSK_IPAR_WRITE_BAS_VARIABLES`

  Description:
  Controls whether the variables section is written to the basic solution file.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `write_data_compressed`

  Corresponding constant:
  `MSK_IPAR_WRITE_DATA_COMPRESSED`

  Description:
  Controls whether the data file is compressed while it is written. 0 means no compression while higher values mean more compression.

  Possible Values:
  Any number between 0 and +inf.

Default value:
0

- `write_data_format`

Corresponding constant:
`MSK_IPAR_WRITE_DATA_FORMAT`

Description:

Controls the file format when writing task data to a file.

Possible Values:

`MSK_DATA_FORMAT_XML` The data file is an XML formatted file.

`MSK_DATA_FORMAT_EXTENSION` The file extension is used to determine the data file format.

`MSK_DATA_FORMAT_MPS` The data file is MPS formatted.

`MSK_DATA_FORMAT_LP` The data file is LP formatted.

`MSK_DATA_FORMAT_MBT` The data file is a MOSEK binary task file.

`MSK_DATA_FORMAT_OP` The data file is an optimization problem formatted file.

Default value:
MSK_DATA_FORMAT_EXTENSION

- `write_data_param`

Corresponding constant:
`MSK_IPAR_WRITE_DATA_PARAM`

Description:
If this option is turned on the parameter settings are written to the data file as parameters.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `write_free_con`

Corresponding constant:
`MSK_IPAR_WRITE_FREE_CON`

Description:
Controls whether the free constraints are written to the data file.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `write_generic_names`

  Corresponding constant:
  `MSK_IPAR_WRITE_GENERIC_NAMES`

  Description:
  Controls whether the generic names or user-defined names are used in the data file.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_OFF

- `write_generic_names_io`

  Corresponding constant:
  `MSK_IPAR_WRITE_GENERIC_NAMES_IO`

  Description:
  Index origin used in generic names.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  1

- `write_int_constraints`

  Corresponding constant:
  `MSK_IPAR_WRITE_INT_CONSTRAINTS`

  Description:
  Controls whether the constraint section is written to the integer solution file.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `write_int_head`

  Corresponding constant:
  `MSK_IPAR_WRITE_INT_HEAD`

  Description:
  Controls whether the header section is written to the integer solution file.

  Possible Values:

  `MSK_ON` Switch the option on.
  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `write_int_variables`

  Corresponding constant:
  `MSK_IPAR_WRITE_INT_VARIABLES`

  Description:
  Controls whether the variables section is written to the integer solution file.

  Possible Values:

  `MSK_ON` Switch the option on.
  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `write_lp_line_width`

  Corresponding constant:
  `MSK_IPAR_WRITE_LP_LINE_WIDTH`

  Description:
  Maximum width of line in an LP file written by MOSEK.

  Possible Values:
  Any positive number.

  Default value:
  80

- `write_lp_quoted_names`

Corresponding constant:
`MSK_IPAR_WRITE_LP_QUOTED_NAMES`

Description:
If this option is turned on, then MOSEK will quote invalid LP names when writing an LP file.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `write_lp_strict_format`

Corresponding constant:
`MSK_IPAR_WRITE_LP_STRICT_FORMAT`

Description:
Controls whether LP output files satisfy the LP format strictly.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `write_lp_terms_per_line`

Corresponding constant:
`MSK_IPAR_WRITE_LP_TERMS_PER_LINE`

Description:
Maximum number of terms on a single line in an LP file written by MOSEK. 0 means unlimited.

Possible Values:
Any number between 0 and +inf.

Default value:
10

- `write_mps_int`

Corresponding constant:
`MSK_IPAR_WRITE_MPS_INT`

Description:
Controls if marker records are written to the MPS file to indicate whether variables are integer restricted.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `write_mps_obj_sense`

Corresponding constant:
`MSK_IPAR_WRITE_MPS_OBJ_SENSE`

Description:
If turned off, the objective sense section is not written to the MPS file.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `write_mps_quoted_names`

Corresponding constant:
`MSK_IPAR_WRITE_MPS_QUOTED_NAMES`

Description:
If a name contains spaces (blanks) when writing an MPS file, then the quotes will be removed.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_ON

- `write_mps_strict`

Corresponding constant:
`MSK_IPAR_WRITE_MPS_STRICT`

Description:
Controls whether the written MPS file satisfies the MPS format strictly or not.

Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:
MSK_OFF

- `write_precision`

  Corresponding constant:
  `MSK_IPAR_WRITE_PRECISION`

  Description:
  Controls the precision with which `double` numbers are printed in the MPS data file. In general it is not worthwhile to use a value higher than 15.

  Possible Values:
  Any number between 0 and +inf.

  Default value:
  8

- `write_sol_constraints`

  Corresponding constant:
  `MSK_IPAR_WRITE_SOL_CONSTRAINTS`

  Description:
  Controls whether the constraint section is written to the solution file.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:
  MSK_ON

- `write_sol_head`

  Corresponding constant:
  `MSK_IPAR_WRITE_SOL_HEAD`

  Description:
  Controls whether the header section is written to the solution file.

  Possible Values:

`MSK_ON` Switch the option on.

`MSK_OFF` Switch the option off.

Default value:

MSK_ON

- `write_sol_variables`

  Corresponding constant:

  `MSK_IPAR_WRITE_SOL_VARIABLES`

  Description:

  Controls whether the variables section is written to the solution file.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:

  MSK_ON

- `write_task_inc_sol`

  Corresponding constant:

  `MSK_IPAR_WRITE_TASK_INC_SOL`

  Description:

  Controls whether the solutions are stored in the task file too.

  Possible Values:

  `MSK_ON` Switch the option on.

  `MSK_OFF` Switch the option off.

  Default value:

  MSK_ON

- `write_xml_mode`

  Corresponding constant:

  `MSK_IPAR_WRITE_XML_MODE`

  Description:

  Controls if linear coefficients should be written by row or column when writing in the XML file format.

  Possible Values:

  `MSK_WRITE_XML_MODE_COL` Write in column order.

  `MSK_WRITE_XML_MODE_ROW` Write in row order.

  Default value:

  MSK_WRITE_XML_MODE_ROW

C.4 String parameter types

- MSK_SPAR_BAS_SOL_FILE_NAME 333
  Name of the bas solution file.
- MSK_SPAR_DATA_FILE_NAME 333
  Data are read and written to this file.
- MSK_SPAR_DEBUG_FILE_NAME 334
  MOSEK debug file.
- MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL 334
  Feasibility repair name violation name.
- MSK_SPAR_INT_SOL_FILE_NAME 334
  Name of the int solution file.
- MSK_SPAR_ITR_SOL_FILE_NAME 334
  Name of the itr solution file.
- MSK_SPAR_PARAM_COMMENT_SIGN 335
  Solution file comment character.
- MSK_SPAR_PARAM_READ_FILE_NAME 335
  Modifications to the parameter database is read from this file.
- MSK_SPAR_PARAM_WRITE_FILE_NAME 335
  The parameter database is written to this file.
- MSK_SPAR_READ_MPS_BOU_NAME 335
  Name of the BOUNDS vector used. An empty name means that the first BOUNDS vector is used.
- MSK_SPAR_READ_MPS_OBJ_NAME 336
  Objective name in the MPS file.
- MSK_SPAR_READ_MPS_RAN_NAME 336
  Name of the RANGE vector used. An empty name means that the first RANGE vector is used.
- MSK_SPAR_READ_MPS_RHS_NAME 336
  Name of the RHS used. An empty name means that the first RHS vector is used.
- MSK_SPAR_SOL_FILTER_XC_LOW 337
  Solution file filter.

- `MSK_SPAR_SOL_FILTER_XC_UPR` 337
  Solution file filter.

- `MSK_SPAR_SOL_FILTER_XX_LOW` 337
  Solution file filter.

- `MSK_SPAR_SOL_FILTER_XX_UPR` 338
  Solution file filter.

- `MSK_SPAR_STAT_FILE_NAME` 338
  Statistics file name.

- `MSK_SPAR_STAT_KEY` 338
  Key used when writing the summary file.

- `MSK_SPAR_STAT_NAME` 339
  Name used when writing the statistics file.

- `MSK_SPAR_WRITE_LP_GEN_VAR_NAME` 339
  Added variable names in the LP files.

- `bas_sol_file_name`

  Corresponding constant:
  `MSK_SPAR_BAS_SOL_FILE_NAME`

  Description:
  Name of the `bas` solution file.

  Possible Values:
  Any valid file name.

  Default value:
  ""

- `data_file_name`

  Corresponding constant:
  `MSK_SPAR_DATA_FILE_NAME`

  Description:
  Data are read and written to this file.

  Possible Values:
  Any valid file name.

  Default value:
  ""

- `debug_file_name`

  Corresponding constant:
  `MSK_SPAR_DEBUG_FILE_NAME`

  Description:
  MOSEK debug file.

  Possible Values:
  Any valid file name.

  Default value:
  ""

- `feasrepair_name_wsumviol`

  Corresponding constant:
  `MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL`

  Description:
  The constraint and variable associated with the total weighted sum of violations are each given the name of this parameter postfixed with `CON` and `VAR` respectively.

  Possible Values:
  Any valid string.

  Default value:
  "WSUMVIOL"

- `int_sol_file_name`

  Corresponding constant:
  `MSK_SPAR_INT_SOL_FILE_NAME`

  Description:
  Name of the `int` solution file.

  Possible Values:
  Any valid file name.

  Default value:
  ""

- `itr_sol_file_name`

  Corresponding constant:
  `MSK_SPAR_ITR_SOL_FILE_NAME`

  Description:
  Name of the `itr` solution file.

Possible Values:
Any valid file name.

Default value:
""

- `param_comment_sign`

  Corresponding constant:
  `MSK_SPAR_PARAM_COMMENT_SIGN`

  Description:
  Only the first character in this string is used. It is considered as a start of comment sign in the MOSEK parameter file. Spaces are ignored in the string.

  Possible Values:
  Any valid string.

  Default value:
  "%%"

- `param_read_file_name`

  Corresponding constant:
  `MSK_SPAR_PARAM_READ_FILE_NAME`

  Description:
  Modifications to the parameter database is read from this file.

  Possible Values:
  Any valid file name.

  Default value:
  ""

- `param_write_file_name`

  Corresponding constant:
  `MSK_SPAR_PARAM_WRITE_FILE_NAME`

  Description:
  The parameter database is written to this file.

  Possible Values:
  Any valid file name.

  Default value:
  ""

- `read_mps_bou_name`

Corresponding constant:
`MSK_SPAR_READ_MPS_BOU_NAME`

Description:
Name of the BOUNDS vector used. An empty name means that the first BOUNDS vector is used.

Possible Values:
Any valid MPS name.

Default value:
""

- `read_mps_obj_name`

Corresponding constant:
`MSK_SPAR_READ_MPS_OBJ_NAME`

Description:
Name of the free constraint used as objective function. An empty name means that the first constraint is used as objective function.

Possible Values:
Any valid MPS name.

Default value:
""

- `read_mps_ran_name`

Corresponding constant:
`MSK_SPAR_READ_MPS_RAN_NAME`

Description:
Name of the RANGE vector used. An empty name means that the first RANGE vector is used.

Possible Values:
Any valid MPS name.

Default value:
""

- `read_mps_rhs_name`

Corresponding constant:
`MSK_SPAR_READ_MPS_RHS_NAME`

Description:
Name of the RHS used. An empty name means that the first RHS vector is used.

Possible Values:
Any valid MPS name.

Default value:
""

- `sol_filter_xc_low`

  Corresponding constant:
  `MSK_SPAR_SOL_FILTER_XC_LOW`

  Description:
  A filter used to determine which constraints should be listed in the solution file. A value of "0.5" means that all constraints having `xc[i]>0.5` should be listed, whereas "+0.5" means that all constraints having `xc[i]>=blc[i]+0.5` should be listed. An empty filter means that no filter is applied.

  Possible Values:
  Any valid filter.

  Default value:
  ""

- `sol_filter_xc_upr`

  Corresponding constant:
  `MSK_SPAR_SOL_FILTER_XC_UPR`

  Description:
  A filter used to determine which constraints should be listed in the solution file. A value of "0.5" means that all constraints having `xc[i]<0.5` should be listed, whereas "-0.5" means all constraints having `xc[i]<=buc[i]-0.5` should be listed. An empty filter means that no filter is applied.

  Possible Values:
  Any valid filter.

  Default value:
  ""

- `sol_filter_xx_low`

  Corresponding constant:
  `MSK_SPAR_SOL_FILTER_XX_LOW`

  Description:
  A filter used to determine which variables should be listed in the solution file. A value of "0.5" means that all constraints having `xx[j]>=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:
""

- `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:
  ""

- `stat_file_name`

  Corresponding constant:
  `MSK_SPAR_STAT_FILE_NAME`

  Description:
  Statistics file name.

  Possible Values:
  Any valid file name.

  Default value:
  ""

- `stat_key`

  Corresponding constant:
  `MSK_SPAR_STAT_KEY`

  Description:
  Key used when writing the summary file.

  Possible Values:
  Any valid XML string.

  Default value:
  ""

- `stat_name`

  Corresponding constant:
  `MSK_SPAR_STAT_NAME`

  Description:
  Name used when writing the statistics file.

  Possible Values:
  Any valid XML string.

  Default value:
  ""

- `write_lp_gen_var_name`

  Corresponding constant:
  `MSK_SPAR_WRITE_LP_GEN_VAR_NAME`

  Description:
  Sometimes when an LP file is written additional variables must be inserted. They will have the prefix denoted by this parameter.

  Possible Values:
  Any valid string.

  Default value:
  "xmskgen"

Appendix D

Symbolic constants

D.1 Constraint or variable access modes

Value	Name Description
0	`MSK_ACC_VAR` Access data by columns (variable orinted)
1	`MSK_ACC_CON` Access data by rows (constraint oriented)

D.2 Basis identification

Value	Name Description
1	`MSK_BI_ALWAYS` Basis identification is always performed even if the interior-point optimizer terminates abnormally.
2	`MSK_BI_NO_ERROR` Basis identification is performed if the interior-point optimizer terminates without an error.
0	`MSK_BI_NEVER` Never do basis identification.
3	`MSK_BI_IF_FEASIBLE` Basis identification is not performed if the interior-point optimizer terminates with a problem status saying that the problem is primal or dual infeasible.

continued on next page

continued from previous page

4	`MSK_BI_OTHER` Try another BI method.

D.3 Bound keys

Value	Name Description
2	`MSK_BK_FX` The constraint or variable is fixed.
0	`MSK_BK_LO` The constraint or variable has a finite lower bound and an infinite upper bound.
3	`MSK_BK_FR` The constraint or variable is free.
1	`MSK_BK_UP` The constraint or variable has an infinite lower bound and an finite upper bound.
4	`MSK_BK_RA` The constraint or variable is ranged.

D.4 Specifies the branching direction.

Value	Name Description
2	`MSK_BRANCH_DIR_DOWN` The mixed integer optimizer always chooses the down branch first.
1	`MSK_BRANCH_DIR_UP` The mixed integer optimizer always chooses the up branch first.
0	`MSK_BRANCH_DIR_FREE` The mixed optimizer decides which branch to choose.

D.5 Progress call-back codes

continued on next page

continued from previous page

Value	Name Description
17	`MSK_CALLBACK_BEGIN_PRIMAL_SENSITIVITY` Primal sensitivity analysis is started.
70	`MSK_CALLBACK_NEW_INT_MIO` The call-back function is called after a new integer solution has been located by the mixed integer optimizer.
37	`MSK_CALLBACK_END_NETWORK_PRIMAL_SIMPLEX` The call-back function is called when the primal network simplex optimizer is terminated.
79	`MSK_CALLBACK_UPDATE_PRESOLVE` The call-back function is called from within the presolve procedure.
55	`MSK_CALLBACK_IM_LICENSE_WAIT` MOSEK is waiting for a license.
1	`MSK_CALLBACK_BEGIN_CONCURRENT` Concurrent optimizer is started.
76	`MSK_CALLBACK_UPDATE_NETWORK_DUAL_SIMPLEX` The call-back function is called in the dual network simplex optimizer.
48	`MSK_CALLBACK_IGNORE_VALUE` This code means that the call-back does not indicate a new phase in the optimization, but is simply a time-triggered call-back.
46	`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.
42	`MSK_CALLBACK_END_PRIMAL_SENSITIVITY` Primal sensitivity analysis is terminated.
20	`MSK_CALLBACK_BEGIN_SIMPLEX` The call-back function is called when the simplex optimizer is started.
40	`MSK_CALLBACK_END_PRESOLVE` The call-back function is called when the presolve is completed.
22	`MSK_CALLBACK_BEGIN_SIMPLEX_NETWORK_DETECT` The call-back function is called when the network detection procedure is started.
13	`MSK_CALLBACK_BEGIN_NETWORK_SIMPLEX` The call-back function is called when the simplex network optimizer is started.
35	`MSK_CALLBACK_END_MIO`

continued on next page

continued from previous page

	The call-back function is called when the mixed integer optimizer is terminated.
73	`MSK_CALLBACK_QCONE` The call-back function is called from within the Qcone optimizer.
27	`MSK_CALLBACK_END_CONIC` The call-back function is called when the conic optimizer is terminated.
11	`MSK_CALLBACK_BEGIN_NETWORK_DUAL_SIMPLEX` The call-back function is called when the dual network simplex optimizer is started.
7	`MSK_CALLBACK_BEGIN_INFEAS_ANA` The call-back function is called when the infeasibility analyzer is started.
67	`MSK_CALLBACK_IM_PRIMAL_SIMPLEX` The call-back function is called at an intermediate point in the primal simplex optimizer.
63	`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.
61	`MSK_CALLBACK_IM_NETWORK_DUAL_SIMPLEX` The call-back function is called at an intermediate point in the dual network simplex optimizer.
66	`MSK_CALLBACK_IM_PRIMAL_SENSIVITY` The call-back function is called at an intermediate stage of the primal sensitivity analysis.
38	`MSK_CALLBACK_END_NETWORK_SIMPLEX` The call-back function is called when the simplex network optimizer is terminated.
34	`MSK_CALLBACK_END_LICENSE_WAIT` End waiting for license.
28	`MSK_CALLBACK_END_DUAL_BI` The call-back function is called from within the basis identification procedure when the dual phase is terminated.
29	`MSK_CALLBACK_END_DUAL_SENSITIVITY` Dual sensitivity analysis is terminated.
24	`MSK_CALLBACK_DUAL_SIMPLEX` The call-back function is called from within the dual simplex optimizer.

continued on next page

continued from previous page

21	`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.
8	`MSK_CALLBACK_BEGIN_INTPNT` The call-back function is called when the interior-point optimizer is started.
52	`MSK_CALLBACK_IM_DUAL_SENSIVITY` The call-back function is called at an intermediate stage of the dual sensitivity analysis.
47	`MSK_CALLBACK_END_SIMPLEX_NETWORK_DETECT` The call-back function is called when the network detection procedure is terminated.
45	`MSK_CALLBACK_END_SIMPLEX` The call-back function is called when the simplex optimizer is terminated.
72	`MSK_CALLBACK_PRIMAL_SIMPLEX` The call-back function is called from within the primal simplex optimizer.
26	`MSK_CALLBACK_END_CONCURRENT` Concurrent optimizer is terminated.
56	`MSK_CALLBACK_IM_MIO` The call-back function is called at an intermediate point in the mixed integer optimizer.
31	`MSK_CALLBACK_END_DUAL_SIMPLEX` The call-back function is called when the dual simplex optimizer is terminated.
36	`MSK_CALLBACK_END_NETWORK_DUAL_SIMPLEX` The call-back function is called when the dual network simplex optimizer is terminated.
54	`MSK_CALLBACK_IM_INTPNT` The call-back function is called at an intermediate stage within the interior-point optimizer where the information database has not been updated.
68	`MSK_CALLBACK_IM_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.
51	`MSK_CALLBACK_IM_DUAL_BI`

continued on next page

continued from previous page

	The call-back function is called from within the basis identification procedure at an intermediate point in the dual phase.
75	`MSK_CALLBACK_UPDATE_DUAL_SIMPLEX` The call-back function is called in the dual simplex optimizer.
82	`MSK_CALLBACK_UPDATE_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.
19	`MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX` The call-back function is called when the primal simplex optimizer is started.
58	`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.
6	`MSK_CALLBACK_BEGIN_DUAL_SIMPLEX` The call-back function is called when the dual simplex optimizer started.
64	`MSK_CALLBACK_IM_PRESOLVE` The call-back function is called from within the presolve procedure at an intermediate stage.
30	`MSK_CALLBACK_END_DUAL_SETUP_BI` The call-back function is called when the dual BI phase is terminated.
3	`MSK_CALLBACK_BEGIN_DUAL_BI` The call-back function is called from within the basis identification procedure when the dual phase is started.
4	`MSK_CALLBACK_BEGIN_DUAL_SENSITIVITY` Dual sensitivity analysis is started.
50	`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.
60	`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.
77	`MSK_CALLBACK_UPDATE_NETWORK_PRIMAL_SIMPLEX` The call-back function is called in the primal network simplex optimizer.
59	`MSK_CALLBACK_IM_MIO_PRESOLVE`

continued on next page

continued from previous page

	The call-back function is called at an intermediate point in the mixed integer optimizer while running the presolve.
14	`MSK_CALLBACK_BEGIN_NONCONVEX` The call-back function is called when the nonconvex optimizer is started.
0	`MSK_CALLBACK_BEGIN_BI` The basis identification procedure has been started.
33	`MSK_CALLBACK_END_INTPNT` The call-back function is called when the interior-point optimizer is terminated.
16	`MSK_CALLBACK_BEGIN_PRIMAL_BI` The call-back function is called from within the basis identification procedure when the primal phase is started.
41	`MSK_CALLBACK_END_PRIMAL_BI` The call-back function is called from within the basis identification procedure when the primal phase is terminated.
18	`MSK_CALLBACK_BEGIN_PRIMAL_SETUP_BI` The call-back function is called when the primal BI setup is started.
32	`MSK_CALLBACK_END_INFEAS_ANA` The call-back function is called when the infeasibility analyzer is terminated.
74	`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.
39	`MSK_CALLBACK_END_NONCONVEX` The call-back function is called when the nonconvex optimizer is terminated.
69	`MSK_CALLBACK_INTPNT` The call-back function is called from within the interior-point optimizer after the information database has been updated.
53	`MSK_CALLBACK_IM_DUAL_SIMPLEX` The call-back function is called at an intermediate point in the dual simplex optimizer.
44	`MSK_CALLBACK_END_PRIMAL_SIMPLEX` The call-back function is called when the primal simplex optimizer is terminated.
81	`MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX` The call-back function is called in the primal simplex optimizer.
80	`MSK_CALLBACK_UPDATE_PRIMAL_BI`

continued on next page

continued from previous page

	The call-back function is called from within the basis identification procedure at an intermediate point in the primal phase.
71	`MSK_CALLBACK_NONCOVEX` The call-back function is called from within the nonconvex optimizer after the information database has been updated.
62	`MSK_CALLBACK_IM_NETWORK_PRIMAL_SIMPLEX` The call-back function is called at an intermediate point in the primal network simplex optimizer.
65	`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.
57	`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.
15	`MSK_CALLBACK_BEGIN_PRESOLVE` The call-back function is called when the presolve is started.
23	`MSK_CALLBACK_CONIC` The call-back function is called from within the conic optimizer after the information database has been updated.
49	`MSK_CALLBACK_IM_BI` The call-back function is called from within the basis identification procedure at an intermediate point.
43	`MSK_CALLBACK_END_PRIMAL_SETUP_BI` The call-back function is called when the primal BI setup is terminated.
10	`MSK_CALLBACK_BEGIN_MIO` The call-back function is called when the mixed integer optimizer is started.
12	`MSK_CALLBACK_BEGIN_NETWORK_PRIMAL_SIMPLEX` The call-back function is called when the primal network simplex optimizer is started.
2	`MSK_CALLBACK_BEGIN_CONIC` The call-back function is called when the conic optimizer is started.
9	`MSK_CALLBACK_BEGIN_LICENSE_WAIT` Begin waiting for license.
25	`MSK_CALLBACK_END_BI` The call-back function is called when the basis identification procedure is terminated.
78	`MSK_CALLBACK_UPDATE_NONCONVEX`

continued on next page

continued from previous page

Value	Name Description
	The call-back function is called at an intermediate stage within the nonconvex optimizer where the information database has been updated.
5	`MSK_CALLBACK_BEGIN_DUAL_SETUP_BI`
	The call-back function is called when the dual BI phase is started.

D.6 Types of convexity checks.

Value	Name Description
1	`MSK_CHECK_CONVEXITY_SIMPLE`
	Perform simple and fast convexity check.
0	`MSK_CHECK_CONVEXITY_NONE`
	No convexity check.

D.7 Compression types

Value	Name Description
2	`MSK_COMPRESS_GZIP`
	The type of compression used is gzip compatible.
0	`MSK_COMPRESS_NONE`
	No compression is used.
1	`MSK_COMPRESS_FREE`
	The type of compression used is chosen automatically.

D.8 Cone types

Value	Name Description
0	`MSK_CT_QUAD`
	The cone is a quadratic cone.
1	`MSK_CT_RQUAD`
	The cone is a rotated quadratic cone.

D.9 CPU type

Value	Name	Description
8	`MSK_CPU_POWERPC_G5`	A G5 PowerPC CPU.
9	`MSK_CPU_INTEL_PM`	An Intel PM cpu.
1	`MSK_CPU_GENERIC`	An generic CPU type for the platform
0	`MSK_CPU_UNKNOWN`	An unknown CPU.
7	`MSK_CPU_AMD_OPTERON`	An AMD Opteron (64 bit).
6	`MSK_CPU_INTEL_ITANIUM2`	An Intel Itanium2.
4	`MSK_CPU_AMD_ATHLON`	An AMD Athlon.
5	`MSK_CPU_HP_PARISC20`	An HP PA RISC version 2.0 CPU.
3	`MSK_CPU_INTEL_P4`	An Intel Pentium P4 or Intel Xeon.
2	`MSK_CPU_INTEL_P3`	An Intel Pentium P3.
10	`MSK_CPU_INTEL_CORE2`	An Intel CORE2 cpu.

D.10 Data format types

Value	Name	Description
5	`MSK_DATA_FORMAT_XML`	The data file is an XML formatted file.
0	`MSK_DATA_FORMAT_EXTENSION`	The file extension is used to determine the data file format.
1	`MSK_DATA_FORMAT_MPS`	The data file is MPS formatted.
2	`MSK_DATA_FORMAT_LP`	

continued on next page

continued from previous page

	The data file is LP formatted.
3	`MSK_DATA_FORMAT_MBT` The data file is a MOSEK binary task file.
4	`MSK_DATA_FORMAT_OP` The data file is an optimization problem formatted file.

D.11 Double information items

Value	Name Description
12	`MSK_DINF_INTPNT_PRIMAL_FEAS` Primal feasibility measure reported by the interior-point or Qcone optimizers. (For the interior-point optimizer this measure does not directly related to the original problem because a homogeneous model is employed).
11	`MSK_DINF_INTPNT_ORDER_CPUTIME` Order time (in CPU seconds).
24	`MSK_DINF_PRESOLVE_CPUTIME` Total time (in CPU seconds) spent in the presolve since it was invoked.
27	`MSK_DINF_RD_CPUTIME` Time (in CPU seconds) spent reading the data file.
28	`MSK_DINF_SIM_CPUTIME` Time (in CPU seconds) spent in the simplex optimizer since invoking it.
32	`MSK_DINF_SOL_BAS_MAX_DBI` Maximal dual bound infeasibility in the basic solution. Updated at the end of the optimization.
47	`MSK_DINF_SOL_ITR_MAX_PCNI` Maximal primal cone infeasibility in the interior-point solution. Updated at the end of the optimization.
19	`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`.
4	`MSK_DINF_CONCURRENT_CPUTIME` Time (in CPU seconds) spent within the concurrent optimizer since its invocation.
49	`MSK_DINF_SOL_ITR_MAX_PINTI`

continued on next page

continued from previous page

	Maximal primal integer infeasibility in the interior-point solution. Updated at the end of the optimization.
30	`MSK_DINF_SIM_OBJ` Objective value reported by the simplex optimizer.
20	`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 $\frac{\lvert(\text{objective value of feasible solution}) - (\text{objective bound})\rvert}{\max(1, \lvert(\text{objective value of feasible solution})\rvert)}.$ Otherwise it has the value -1.0.
37	`MSK_DINF_SOL_BAS_PRIMAL_OBJ` Primal objective value of the basic solution. Updated at the end of the optimization.
26	`MSK_DINF_PRESOLVE_LINDEP_CPUTIME` Total time (in CPU seconds) spent in the linear dependency checker since the presolve was invoked.
46	`MSK_DINF_SOL_ITR_MAX_PBI` Maximal primal bound infeasibility in the interior-point solution. Updated at the end of the optimization.
34	`MSK_DINF_SOL_BAS_MAX_PBI` Maximal primal bound infeasibility in the basic solution. Updated at the end of the optimization.
44	`MSK_DINF_SOL_ITR_MAX_DCNI` Maximal dual cone infeasibility in the interior-point solution. Updated at the end of the optimization.
8	`MSK_DINF_INTPNT_DUAL_OBJ` Dual objective value reported by the interior-point or Qcone optimizer.
45	`MSK_DINF_SOL_ITR_MAX_DEQI` Maximal dual equality infeasibility in the interior-point solution. Updated at the end of the optimization.
14	`MSK_DINF_INTPNT_REALTIME` Time (in wall-clock seconds) spent within the interior-point optimizer since its invocation.
29	`MSK_DINF_SIM_FEAS` Feasibility measure reported by the simplex optimizer.

continued on next page

continued from previous page

15	`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.
36	`MSK_DINF_SOL_BAS_MAX_PINTI` Maximal primal integer infeasibility in the basic solution. Updated at the end of the optimization.
40	`MSK_DINF_SOL_INT_MAX_PINTI` Maximal primal integer infeasibility in the integer solution. Updated at the end of the optimization.
6	`MSK_DINF_INTPNT_CPUTIME` Time (in CPU seconds) spent within the interior-point optimizer since its invocation.
35	`MSK_DINF_SOL_BAS_MAX_PEQI` Maximal primal equality infeasibility in the basic solution. Updated at the end of the optimization.
9	`MSK_DINF_INTPNT_FACTOR_NUM_FLOPS` An estimate of the number of flops used in the factorization.
42	`MSK_DINF_SOL_ITR_DUAL_OBJ` Dual objective value of the interior-point solution. Updated at the end of the optimization.
22	`MSK_DINF_OPTIMIZER_CPUTIME` Total time (in CPU seconds) spent in the optimizer since it was invoked.
38	`MSK_DINF_SOL_INT_MAX_PBI` Maximal primal bound infeasibility in the integer solution. Updated at the end of the optimization.
7	`MSK_DINF_INTPNT_DUAL_FEAS` Dual feasibility measure reported by the interior-point and Qcone optimizer. (For the interior-point optimizer this measure does not directly related to the original problem because a homogeneous model is employed.)
23	`MSK_DINF_OPTIMIZER_REALTIME` Total time (in wall-clock seconds) spent in the optimizer since it was invoked.
10	`MSK_DINF_INTPNT_KAP_DIV_TAU`

continued on next page

continued from previous page

	This measure should converge to zero if the problem has a primal-dual optimal solution or to infinity if problem is (strictly) primal or dual infeasible. In case the measure is converging towards a positive but bounded constant the problem is usually ill-posed.
3	`MSK_DINF_BI_PRIMAL_CPUTIME` Time (in CPU seconds) spent within the primal phase of the basis identification procedure since its invocation.
39	`MSK_DINF_SOL_INT_MAX_PEQI` Maximal primal equality infeasibility in the basic solution. Updated at the end of the optimization.
50	`MSK_DINF_SOL_ITR_PRIMAL_OBJ` Primal objective value of the interior-point solution. Updated at the end of the optimization.
21	`MSK_DINF_MIO_USER_OBJ_CUT` If the objective cut is used, then this information item has the value of the cut.
25	`MSK_DINF_PRESOLVE_ELI_CPUTIME` Total time (in CPU seconds) spent in the eliminator since the presolve was invoked.
5	`MSK_DINF_CONCURRENT_REALTIME` Time (in wall-clock seconds) within the concurrent optimizer since its invocation.
18	`MSK_DINF_MIO_OBJ_BOUND` The best bound objective value corresponding to the best integer feasible solution is located. Please note that at least one integer feasible solution must be located i.e. check `MSK_IINF_MIO_NUM_INT_SOLUTIONS`.
1	`MSK_DINF_BI_CPUTIME` Time (in CPU seconds) spent within the basis identification procedure since its invocation.
31	`MSK_DINF_SOL_BAS_DUAL_OBJ` Dual objective value of the basic solution. Updated at the end of the optimization.
13	`MSK_DINF_INTPNT_PRIMAL_OBJ` Primal objective value reported by the interior-point or Qcone optimizer.
41	`MSK_DINF_SOL_INT_PRIMAL_OBJ` Primal objective value of the integer solution. Updated at the end of the optimization.

continued on next page

continued from previous page

16	`MSK_DINF_MIO_CPUTIME` Time spent in the mixed integer optimizer.
17	`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.
33	`MSK_DINF_SOL_BAS_MAX_DEQI` Maximal dual equality infeasibility in the basic solution. Updated at the end of the optimization.
48	`MSK_DINF_SOL_ITR_MAX_PEQI` Maximal primal equality infeasibility in the interior-point solution. Updated at the end of the optimization.
2	`MSK_DINF_BI_DUAL_CPUTIME` Time (in CPU seconds) spent within the dual phase basis identification procedure since its invocation.
0	`MSK_DINF_BI_CLEAN_CPUTIME` Time (in CPU seconds) spent within the clean-up phase of the basis identification procedure since its invocation.
43	`MSK_DINF_SOL_ITR_MAX_DBI` Maximal dual bound infeasibility in the interior-point solution. Updated at the end of the optimization.

D.12 Double parameters

Value	Name Description
38	`MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH` If the lower objective cut is less than the value of this parameter value, then the lower objective cut i.e. `MSK_DPAR_LOWER_OBJ_CUT` is treated as $-\infty$.
41	`MSK_DPAR_MIO_MAX_TIME` This parameter limits the maximum time spent by the mixed integer optimizer. A negative number means infinity.
1	`MSK_DPAR_BASIS_TOL_S`

continued on next page

continued from previous page

	Maximum absolute dual bound violation in an optimal basic solution.
56	`MSK_DPAR_PRESOLVE_TOL_S` Absolute zero tolerance employed for s_i in the presolve.
59	`MSK_DPAR_UPPER_OBJ_CUT` If a feasible solution having and objective value outside, the interval [`MSK_DPAR_LOWER_OBJ_CUT`, `MSK_DPAR_UPPER_OBJ_CUT`], then MOSEK is terminated.
14	`MSK_DPAR_INTPNT_CO_TOL_DFEAS` Dual feasibility tolerance used by the conic interior-point optimizer.
6	`MSK_DPAR_DATA_TOL_AIJ_LARGE` An element in A which is larger than this value in absolute size causes a warning message to be printed.
46	`MSK_DPAR_MIO_TOL_ABS_GAP` Absolute optimality tolerance employed by the mixed integer optimizer.
60	`MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH` 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 ∞.
52	`MSK_DPAR_NONCONVEX_TOL_OPT` Optimality tolerance used by the nonconvex optimizer.
51	`MSK_DPAR_NONCONVEX_TOL_FEAS` Feasibility tolerance used by the nonconvex optimizer.
40	`MSK_DPAR_MIO_HEURISTIC_TIME` 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.
57	`MSK_DPAR_PRESOLVE_TOL_X` Absolute zero tolerance employed for x_j in the presolve.
22	`MSK_DPAR_INTPNT_NL_TOL_MU_RED` Relative complementarity gap tolerance.
44	`MSK_DPAR_MIO_NEAR_TOL_REL_GAP` The mixed integer optimizer is terminated when this tolerance is satisfied. This termination criteria is delayed. See `MSK_DPAR_MIO_DISABLE_TERM_TIME` for details.
58	`MSK_DPAR_SIMPLEX_ABS_TOL_PIV` Absolute pivot tolerance employed by the simplex optimizers.
5	`MSK_DPAR_DATA_TOL_AIJ` Absolute zero tolerance for elements in A.

continued on next page

continued from previous page

13	`MSK_DPAR_FEASREPAIR_TOL` Tolerance for constraint enforcing upper bound on sum of weighted violations in feasibility repair.
28	`MSK_DPAR_INTPNT_TOL_DSAFE` Controls the initial dual starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly.
29	`MSK_DPAR_INTPNT_TOL_INFEAS` Controls when the optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.
23	`MSK_DPAR_INTPNT_NL_TOL_NEAR_REL` If the MOSEK nonlinear interior-point optimizer cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.
53	`MSK_DPAR_OPTIMIZER_MAX_TIME` Maximum amount of time the optimizer is allowed to spent on the optimization. A negative number means infinity.
12	`MSK_DPAR_DATA_TOL_X` Zero tolerance for constraints and variables i.e. if the distance between the lower and upper bound is less than this value, then the lower and lower bound is considered identical.
45	`MSK_DPAR_MIO_REL_ADD_CUT_LIMITED` Controls how many cuts the mixed integer optimizer is allowed to add to the problem. Let α be the value of this parameter and m the number constraints, then mixed integer optimizer is allowed to αm cuts.
3	`MSK_DPAR_BI_LU_TOL_REL_PIV` Relative pivot tolerance used in the LU factorization in the basis identification procedure.
30	`MSK_DPAR_INTPNT_TOL_MU_RED` Relative complementarity gap tolerance.
16	`MSK_DPAR_INTPNT_CO_TOL_MU_RED` Relative complementarity gap tolerance feasibility tolerance used by the conic interior-point optimizer.
19	`MSK_DPAR_INTPNT_CO_TOL_REL_GAP` Relative gap termination tolerance used by the conic interior-point optimizer.

continued on next page

continued from previous page

37	`MSK_DPAR_LOWER_OBJ_CUT` If a feasible solution having an objective value outside, the interval [MSK_DPAR_LOWER_OBJ_CUT, MSK_DPAR_UPPER_OBJ_CUT], then MOSEK is terminated.
39	`MSK_DPAR_MIO_DISABLE_TERM_TIME` 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 criterias are never checked.
50	`MSK_DPAR_MIO_TOL_X` Absolute solution tolerance used in mixed-integer optimizer.
9	`MSK_DPAR_DATA_TOL_C_HUGE` An element in c which is larger than the value of this parameter in absolute terms is considered to be huge and generates an error.
55	`MSK_DPAR_PRESOLVE_TOL_LIN_DEP` Controls when a constraint is determined to be linearly dependent.
35	`MSK_DPAR_INTPNT_TOL_REL_STEP` Relative step size to the boundary for linear and quadratic optimization problems.
10	`MSK_DPAR_DATA_TOL_CJ_LARGE` An element in c which is larger than this value in absolute terms causes a warning message to be printed.
26	`MSK_DPAR_INTPNT_NL_TOL_REL_STEP` Relative step size to the boundary for general nonlinear optimization problems.
36	`MSK_DPAR_INTPNT_TOL_STEP_SIZE` If the step size falls below the value of this parameter, then the interior-point optimizer assumes it is stalled. It it does not not make any progress.
32	`MSK_DPAR_INTPNT_TOL_PFEAS`

continued on next page

continued from previous page

	Primal feasibility tolerance used for linear and quadratic optimization problems.
0	`MSK_DPAR_BASIS_REL_TOL_S` Maximum relative dual bound violation allowed in an optimal basic solution.
15	`MSK_DPAR_INTPNT_CO_TOL_INFEAS` Controls when the conic interior-point optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.
47	`MSK_DPAR_MIO_TOL_ABS_RELAX_INT` Absolute relaxation tolerance of the integer constraints. I.e. $\min(\lvert x\rvert - \lfloor x\rfloor, \lceil x\rceil - \lvert x\rvert)$ is less than the tolerance then the integer restrictions assumed to be satisfied.
54	`MSK_DPAR_PRESOLVE_TOL_AIJ` Absolute zero tolerance employed for a_{ij} in the presolve.
42	`MSK_DPAR_MIO_MAX_TIME_APRX_OPT` Number of seconds spent by the mixed integer optimizer before the `MSK_DPAR_MIO_TOL_REL_RELAX_INT` is applied.
31	`MSK_DPAR_INTPNT_TOL_PATH` Controls how close the interior-point optimizer follows the central path. A large value of this parameter means the central is followed very closely. On numerical unstable problems it might worthwhile to increase this parameter.
20	`MSK_DPAR_INTPNT_NL_MERIT_BAL` Controls if the complementarity and infeasibility is converging to zero at about equal rates.
2	`MSK_DPAR_BASIS_TOL_X` Maximum absolute primal bound violation allowed in an optimal basic solution.
34	`MSK_DPAR_INTPNT_TOL_REL_GAP` Relative gap termination tolerance.
8	`MSK_DPAR_DATA_TOL_BOUND_WRN` If a bound value is larger than this value in absolute size, then a warning message is issued.
7	`MSK_DPAR_DATA_TOL_BOUND_INF` Any bound which in absolute value is greater than this parameter is considered infinite.
33	`MSK_DPAR_INTPNT_TOL_PSAFE`

continued on next page

continued from previous page

	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 might be worthwhile to increase this value.
17	`MSK_DPAR_INTPNT_CO_TOL_NEAR_REL` If MOSEK cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.
4	`MSK_DPAR_CALLBACK_FREQ` Controls the time between calls to the progress call-back function. Hence, if the value of this parameter is for example 10, then the call-back is called approximately each 10 seconds. A negative value is equivalent to infinity. In general frequent call-backs may hurt the performance.
24	`MSK_DPAR_INTPNT_NL_TOL_PFEAS` Primal feasibility tolerance used when a nonlinear model is solved.
21	`MSK_DPAR_INTPNT_NL_TOL_DFEAS` Dual feasibility tolerance used when a nonlinear model is solved.
48	`MSK_DPAR_MIO_TOL_REL_GAP` Relative optimality tolerance employed by the mixed integer optimizer.
27	`MSK_DPAR_INTPNT_TOL_DFEAS` Dual feasibility tolerance used for linear and quadratic optimization problems.
43	`MSK_DPAR_MIO_NEAR_TOL_ABS_GAP` Relaxed absolute optimality tolerance employed by the mixed integer optimizer. This termination criteria is delayed. See `MSK_DPAR_MIO_DISABLE_TERM_TIME` for details.
49	`MSK_DPAR_MIO_TOL_REL_RELAX_INT` Relative relaxation tolerance of the integer constraints. I.e. $\min(\lvert x\rvert - \lfloor x\rfloor, \lceil x\rceil - \lvert x\rvert)$ is less than the tolerance times $\lvert x\rvert$ then the integer restrictions assumed to be satisfied.
11	`MSK_DPAR_DATA_TOL_QIJ` Absolute zero tolerance for elements in Q matrices.
25	`MSK_DPAR_INTPNT_NL_TOL_REL_GAP` Relative gap termination tolerance for nonlinear problems.
18	`MSK_DPAR_INTPNT_CO_TOL_PFEAS` Primal feasibility tolerance used by the conic interior-point optimizer.

D.13 Double values

Value	Name Description
1e+30	`MSK_INFINITY` Definition of infinity.

D.14 Feasibility repair types

Value	Name Description
0	`MSK_FEASREPAIR_OPTIMIZE_NONE` Do not optimize the feasibility repair problem.
2	`MSK_FEASREPAIR_OPTIMIZE_COMBINED` Minimize with original objective subject to minimal weighted violation of bounds.
1	`MSK_FEASREPAIR_OPTIMIZE_PENALTY` Minimize weighted sum of violations.

D.15 Integer information items.

Value	Name Description
10	`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.
62	`MSK_IINF_SIM_PRIMAL_INF_ITER` The number of iterations taken with primal infeasibility.
45	`MSK_IINF_RD_NUMCON` Number of constraints read.
71	`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.
20	`MSK_IINF_MIO_NUMINT`

continued on next page

continued from previous page

	Number of integer variables in the problem solved be the mixed integer optimizer.
67	MSK_IINF_SOL_INT_PROSTA Problem status of the integer solution. Updated after each optimization.
51	MSK_IINF_RD_PROTYPE Problem type.
73	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	MSK_IINF_MIO_TOTAL_NUM_CLIQUE_CUTS Number of clique cuts.
68	MSK_IINF_SOL_INT_SOLSTA Solution status of the integer solution. Updated after each optimization.
12	MSK_IINF_MIO_NUM_ACTIVE_NODES Number of active nodes in the branch and bound tree.
9	MSK_IINF_INTPNT_SOLVE_DUAL Non-zero if the interior-point optimizer is solving the dual problem.
39	MSK_IINF_MIO_TOTAL_NUM_RELAX Number of relaxations solved during the optimization.
28	MSK_IINF_MIO_TOTAL_NUM_CUTS Total number of cuts generated by the mixed integer optimizer.
2	MSK_IINF_CACHE_SIZE_L2 L2 cache size used.
66	MSK_IINF_SOL_BAS_SOLSTA Solution status of the basic solution. Updated after each optimization.
50	MSK_IINF_RD_NUMVAR Number of variables read.
64	MSK_IINF_SIM_SOLVE_DUAL Is non-zero if dual problem is solved.
47	MSK_IINF_RD_NUMINTVAR Number of integer constrained variables read.
46	MSK_IINF_RD_NUMCONE Number of conic constraints read.
43	MSK_IINF_OPTIMIZE_RESPONSE

continued on next page

continued from previous page

	The reponse code returned by optimize.
52	`MSK_IINF_SIM_DUAL_DEG_ITER` The number of dual degenerate iterations.
69	`MSK_IINF_SOL_ITR_PROSTA` Problem status of the interior-point solution. Updated after each optimization.
37	`MSK_IINF_MIO_TOTAL_NUM_OBJ_CUTS` Number of obj cuts.
11	`MSK_IINF_MIO_INITIAL_SOLUTION` Is non-zero if an initial integer solution is specified.
36	`MSK_IINF_MIO_TOTAL_NUM_LIFT_CUTS` Number of lift cuts.
8	`MSK_IINF_INTPNT_NUM_THREADS` Number of threads that the interior-point optimizer is using.
65	`MSK_IINF_SOL_BAS_PROSTA` Problem status of the basic solution. Updated after each optimization.
54	`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.
70	`MSK_IINF_SOL_ITR_SOLSTA` Solution status of the interior-point solution. Updated after each optimization.
19	`MSK_IINF_MIO_NUMCON` Number of constraints in the problem solved be the mixed integer optimizer.
42	`MSK_IINF_OPT_NUMVAR` Number of variables in the problem solved when the optimizer is called
58	`MSK_IINF_SIM_NUMVAR` Number of variables in the problem solved by the simplex optimizer.
35	`MSK_IINF_MIO_TOTAL_NUM_LATTICE_CUTS` Number of lattice cuts.
63	`MSK_IINF_SIM_PRIMAL_ITER` Number of primal simplex iterations during the last optimization.
33	`MSK_IINF_MIO_TOTAL_NUM_GUB_COVER_CUTS` Number of GUB cover cuts.
44	`MSK_IINF_RD_NUMANZ` Number of non-zeros in A that is read.

continued on next page

continued from previous page

5	MSK_IINF_INTPNT_FACTOR_NUM_NZ Number of non-zeros in factorization.
24	MSK_IINF_MIO_TOTAL_NUM_CARDGUB_CUTS Number of cardgub cuts.
16	MSK_IINF_MIO_NUM_INTPNT_ITER Number of interior-point iterations performed by the mixed-integer optimizer.
60	MSK_IINF_SIM_PRIMAL_HOTSTART If 1 then the primal simplex algorithm is solving from an advance basis.
27	MSK_IINF_MIO_TOTAL_NUM_CONTRA_CUTS Number of contra cuts.
0	MSK_IINF_BI_ITER Number of simplex pivots performed since invoking the basis identification procedure.
13	MSK_IINF_MIO_NUM_BRANCH Number of branches performed during the optimization.
3	MSK_IINF_CONCURRENT_FASTEST_OPTIMIZER The type of the optimizer that finished first in a concurrent optimization.
40	MSK_IINF_MIO_USER_OBJ_CUT If it is non-zero, then the objective cut is used.
32	MSK_IINF_MIO_TOTAL_NUM_GOMORY_CUTS Number of Gomory cuts.
1	MSK_IINF_CACHE_SIZE_L1 L1 cache size used.
29	MSK_IINF_MIO_TOTAL_NUM_DISAGG_CUTS Number of diasagg cuts.
48	MSK_IINF_RD_NUMQ Number of nonempty Q matrices read.
26	MSK_IINF_MIO_TOTAL_NUM_COEF_REDC_CUTS Number of coef. redc. cuts.
6	MSK_IINF_INTPNT_FACTOR_NUM_OFFCOL Number of columns in the constraint matrix (or Jacobian) that has an offending structure.
72	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.
17	MSK_IINF_MIO_NUM_RELAX

continued on next page

continued from previous page

		Number of relaxations solved during the optimization.
38	`MSK_IINF_MIO_TOTAL_NUM_PLAN_LOC_CUTS`	Number of loc cuts.
23	`MSK_IINF_MIO_TOTAL_NUM_BRANCH`	Number of branches performed during the optimization.
49	`MSK_IINF_RD_NUMQNZ`	Number of Q non-zeros.
59	`MSK_IINF_SIM_PRIMAL_DEG_ITER`	The number of primal degenerate iterations.
31	`MSK_IINF_MIO_TOTAL_NUM_GCD_CUTS`	Number of gcd cuts.
53	`MSK_IINF_SIM_DUAL_HOTSTART`	If 1 then the dual simplex algorithm is solving from an advance basis.
18	`MSK_IINF_MIO_NUM_SIMPLEX_ITER`	Number of simplex iterations performed by the mixed-integer optimizer.
56	`MSK_IINF_SIM_DUAL_ITER`	Number of dual simplex iterations during the last optimization.
55	`MSK_IINF_SIM_DUAL_INF_ITER`	The number of iterations taken with dual infeasibility.
30	`MSK_IINF_MIO_TOTAL_NUM_FLOW_COVER_CUTS`	Number of flow cover cuts.
21	`MSK_IINF_MIO_NUMVAR`	Number of variables in the problem solved be the mixed integer optimizer.
15	`MSK_IINF_MIO_NUM_INT_SOLUTIONS`	Number of integer feasible solutions that has been found.
61	`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.
14	`MSK_IINF_MIO_NUM_CUTS`	Number of cuts generated by the mixed integer optimizer.
4	`MSK_IINF_CPU_TYPE`	The type of cpu detected.
7	`MSK_IINF_INTPNT_ITER`	Number of interior-point iterations since invoking the interior-point optimizer.
34	`MSK_IINF_MIO_TOTAL_NUM_KNAPSUR_COVER_CUTS`	Number of knapsack cover cuts.

continued on next page

continued from previous page

41	`MSK_IINF_OPT_NUMCON` Number of constraints in the problem solved when the optimizer is called.
22	`MSK_IINF_MIO_TOTAL_NUM_BASIS_CUTS` Number of basis cuts.
57	`MSK_IINF_SIM_NUMCON` Number of constraints in the problem solved by the simplex optimizer.

D.16 Information item types

Value	Name Description
0	`MSK_INF_DOU_TYPE` Is a double information type.
1	`MSK_INF_INT_TYPE` Is an integer.

D.17 Input/output modes

Value	Name Description
0	`MSK_IOMODE_READ` The file is read-only.
1	`MSK_IOMODE_WRITE` The file is write-only. If the file exists then it is truncated when it is opened. Otherwise it is created when it is opened.
2	`MSK_IOMODE_READWRITE` The file is to read and written.

D.18 Integer parameters

Value	Name Description
157	`MSK_IPAR_SIM_STABILITY_PRIORITY` Controls how high priority the numerical stability should be given.

continued on next page

continued from previous page

115	`MSK_IPAR_READ_ADD_CONE` Additional number of conic constraints that is made room for in the problem.
185	`MSK_IPAR_WRITE_MPS_STRICT` Controls whether the written MPS file satisfies the MPS format strictly or not.
21	`MSK_IPAR_INFEAS_REPORT_AUTO` Controls whether an infeasibility report is automatically produced after the optimization if the problem is primal or dual infeasible.
87	`MSK_IPAR_MIO_NODE_OPTIMIZER` Controls which optimizer is employed at the non-root nodes in the mixed integer optimizer.
109	`MSK_IPAR_PRESOLVE_LEVEL` Currently not used.
117	`MSK_IPAR_READ_ADD_VAR` Additional number of variables that is made room for in the problem.
112	`MSK_IPAR_PRESOLVE_USE` Controls whether the presolve is applied to a problem before it is optimized.
64	`MSK_IPAR_LOG_SENSITIVITY_OPT` Controls the amount of logging from the optimizers employed during the sensitivity analysis. 0 means no logging information is produced.
101	`MSK_IPAR_OPF_WRITE_SOL_ITG` If `MSK_IPAR_OPF_WRITE_SOLUTIONS` is `MSK_ON` and an integer solution is defined, write the integer solution in OPF files.
167	`MSK_IPAR_WRITE_BAS_HEAD` Controls whether the header section is written to the basic solution file.
74	`MSK_IPAR_MIO_BRANCH_PRIORITIES_USE` Controls whether branching priorities are used by the mixed integer optimizer.
78	`MSK_IPAR_MIO_CUT_LEVEL_TREE` 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.
169	`MSK_IPAR_WRITE_DATA_COMPRESSED` Controls whether the data file is compressed while it is written. 0 means no compression while higher values mean more compression.
130	`MSK_IPAR_READ_MPS_RELAX`

continued on next page

continued from previous page

	If this option is turned on, then the relaxation of the MIP will be read.
98	`MSK_IPAR_OPF_WRITE_PARAMETERS` Write a parameter section in an OPF file.
119	`MSK_IPAR_READ_CON` Expected maximum number of constraints to be read. The option is only used by fast MPS and LP file readers.
177	`MSK_IPAR_WRITE_INT_VARIABLES` Controls whether the variables section is written to the integer solution file.
113	`MSK_IPAR_READ_ADD_ANZ` Additional number of non-zeros in A that is made room for in the problem.
32	`MSK_IPAR_INTPNT_ORDER_METHOD` Controls the ordering strategy used by the interior-point optimizer when factorizing the Newton equation system.
102	`MSK_IPAR_OPF_WRITE_SOL_ITR` If `MSK_IPAR_OPF_WRITE_SOLUTIONS` is `MSK_ON` and an interior solution is defined, write the interior solution in OPF files.
63	`MSK_IPAR_LOG_SENSITIVITY` Controls the amount of logging during the sensitivity analysis. 0: Means no logging information is produced. 1: Timing information is printed. 2: Sensitivity results are printed.
133	`MSK_IPAR_READ_QNZ` Expected maximum number of Q non-zeros to be read. The option is used only by MPS and LP file readers.
53	`MSK_IPAR_LOG_INFEAS_ANA` Controls amount of output printed by the infeasibility analyzer procedures. A higher level implies that more information is logged.
152	`MSK_IPAR_SIM_PRIMAL_SELECTION` Controls the choice of the incoming variable, known as the selection strategy, in the primal simplex optimizer.
175	`MSK_IPAR_WRITE_INT_CONSTRAINTS` Controls whether the constraint section is written to the integer solution file.
180	`MSK_IPAR_WRITE_LP_STRICT_FORMAT` Controls whether LP output files satisfy the LP format strictly.
138	`MSK_IPAR_SENSITIVITY_TYPE` Controls which type of sensitivity analysis is to be performed.

continued on next page

continued from previous page

141	`MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION` The dual simplex optimizer can use a so-called restricted selection/pricing strategy to chooses the outgoing variable. Hence, if restricted selection is applied, then the dual simplex optimizer first choose a subset of all the potential outgoing variables. Next, for some time it will choose the outgoing variable only among the subset. From time to time the subset is redefined. A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.
56	`MSK_IPAR_LOG_MIO_FREQ` Controls how frequent the mixed integer optimizer prints the log line. It will print line every time `MSK_IPAR_LOG_MIO_FREQ` relaxations have been solved.
100	`MSK_IPAR_OPF_WRITE_SOL_BAS` If `MSK_IPAR_OPF_WRITE_SOLUTIONS` is `MSK_ON` and a basic solution is defined, include the basic solution in OPF files.
91	`MSK_IPAR_MIO_ROOT_OPTIMIZER` Controls which optimizer is employed at the root node in the mixed integer optimizer.
172	`MSK_IPAR_WRITE_FREE_CON` Controls whether the free constraints are written to the data file.
107	`MSK_IPAR_PRESOLVE_ELIM_FILL` Controls the maximum amount of fill-in that can be created during the elimination phase of the presolve. This parameter times (`numcon`+`numvar`) denotes the amount of fill-in.
93	`MSK_IPAR_NONCONVEX_MAX_ITERATIONS` Maximum number of iterations that can be used by the nonconvex optimizer.
82	`MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER`
173	`MSK_IPAR_WRITE_GENERIC_NAMES` Controls whether the generic names or user-defined names are used in the data file.
165	`MSK_IPAR_WARNING_LEVEL` Warning level.
46	`MSK_IPAR_LOG_BI_FREQ`

continued on next page

continued from previous page

	Controls how frequent the optimizer outputs information about the basis identification and how frequent the user-defined call-back function is called.
61	`MSK_IPAR_LOG_PRESOLVE` Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.
8	`MSK_IPAR_CHECK_CTRL_C` Specifies whether MOSEK should check for `<ctrl>+<c>` key presses. In case it has, then control is returned to the user program. In case a user-defined ctrl-c function is defined then that is used to check for ctrl-c. Otherwise the system procedure `signal` is used.
116	`MSK_IPAR_READ_ADD_QNZ` Additional number of non-zeros in the Q matrices that is made room for in the problem.
187	`MSK_IPAR_WRITE_SOL_CONSTRAINTS` Controls whether the constraint section is written to the solution file.
31	`MSK_IPAR_INTPNT_OFF_COL_TRH` 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.
118	`MSK_IPAR_READ_ANZ` Expected maximum number of A non-zeros to be read. The option is used only by fast MPS and LP file readers.
86	`MSK_IPAR_MIO_MODE` Controls whether the optimizer includes the integer restrictions when solving a (mixed) integer optimization problem.
124	`MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU` If this option is turned on, MOSEK will drop variables that are defined for the first time in the bounds section.
65	`MSK_IPAR_LOG_SIM` Controls amount of output printed by the simplex optimizer. A higher level implies that more information is logged.
58	`MSK_IPAR_LOG_OPTIMIZER` Controls the amount of general optimizer information that is logged.
164	`MSK_IPAR_SOLUTION_CALLBACK` Indicates whether solution call-backs will be performed during the optimization.

continued on next page

continued from previous page

66	`MSK_IPAR_LOG_SIM_FREQ` Controls how frequent the simplex optimizer outputs information about the optimization and how frequent the user-defined call-back function is called.
57	`MSK_IPAR_LOG_NONCONVEX` Controls amount of output printed by the nonconvex optimizer.
17	`MSK_IPAR_FEASREPAIR_OPTIMIZE` Controls which type of feasibility analysis is to be performed.
179	`MSK_IPAR_WRITE_LP_QUOTED_NAMES` If this option is turned on, then MOSEK will quote invalid LP names when writing an LP file.
49	`MSK_IPAR_LOG_FACTOR` If turned on, then the factor log lines are added to the log.
125	`MSK_IPAR_READ_LP_QUOTED_NAMES` If a name is in quotes when reading an LP file, the quotes will be removed.
184	`MSK_IPAR_WRITE_MPS_QUOTED_NAMES` If a name contains spaces (blanks) when writing an MPS file, then the quotes will be removed.
131	`MSK_IPAR_READ_MPS_WIDTH` Controls the maximal number of chars allowed in one line of the MPS file.
59	`MSK_IPAR_LOG_ORDER` If turned on, then factor lines are added to the log.
77	`MSK_IPAR_MIO_CUT_LEVEL_ROOT`

continued on next page

continued from previous page

	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
132	`MSK_IPAR_READ_Q_MODE` Controls how the Q matrices are read from the MPS file.
88	`MSK_IPAR_MIO_NODE_SELECTION` Controls the node selection strategy employed by the mixed integer optimizer.
163	`MSK_IPAR_SOL_READ_WIDTH` Controls the maximal acceptable width of line in the solutions when read by MOSEK.
72	`MSK_IPAR_MAXNUMANZ_DOUBLE_TRH` Whenever MOSEK runs out of storage for the A matrix, it will double the value for `maxnumanz` until `maxnumnza` reaches the value of this parameter. When this threshold is reached it will use a slower increase.
29	`MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS` Maximum number of steps to be used by the iterative refinement of the search direction. A negative value implies that the optimizer Chooses the maximum number of iterative refinement steps.
114	`MSK_IPAR_READ_ADD_CON` Additional number of constraints that is made room for in the problem.
47	`MSK_IPAR_LOG_CONCURRENT` Controls amount of output printed by the concurrent optimizer.
67	`MSK_IPAR_LOG_SIM_MINOR` Currently not in use.

continued on next page

continued from previous page

144	`MSK_IPAR_SIM_MAX_ITERATIONS` Maximum number of iterations that can be used by a simplex optimizer.
27	`MSK_IPAR_INTPNT_MAX_ITERATIONS` Controls the maximum number of iterations allowed in the interior-point optimizer.
15	`MSK_IPAR_CPU_TYPE` Specifies the CPU type. By default MOSEK tries to auto detect the CPU type. Therefore, we recommend to change this parameter only if the auto detection does not work properly.
45	`MSK_IPAR_LOG_BI` Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.
28	`MSK_IPAR_INTPNT_MAX_NUM_COR` Controls the maximum number of correctors allowed by the multiple corrector procedure. A negative value means that MOSEK is making the choice.
178	`MSK_IPAR_WRITE_LP_LINE_WIDTH` Maximum width of line in an LP file written by MOSEK.
162	`MSK_IPAR_SOL_READ_NAME_WIDTH` When a solution is read by MOSEK and some constraint, variable or cone names contain blanks, then a maximum name width much be specified. A negative value implies that no name contain blanks.
40	`MSK_IPAR_LICENSE_DEBUG` This option is used to turn on debugging of the incense manager.
43	`MSK_IPAR_LICENSE_WAIT` If all licenses are in use MOSEK returns with an error code. However, by turning on this parameter MOSEK will wait for an available license.
174	`MSK_IPAR_WRITE_GENERIC_NAMES_IO` Index origin used in generic names.
10	`MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS` The maximum number of simultaneous optimizations that will be started by the concurrent optimizer.
153	`MSK_IPAR_SIM_REFACTOR_FREQ` Controls how frequent the basis is refactorized. The value 0 means that the optimizer determines the best point of refactorization. It is strongly recommended NOT to change this parameter.
142	`MSK_IPAR_SIM_DUAL_SELECTION`

continued on next page

continued from previous page

	Controls the choice of the incoming variable, known as the selection strategy, in the dual simplex optimizer.
156	`MSK_IPAR_SIM_SOLVE_FORM` Controls whether the primal or the dual problem is solved by the primal-/dual- simplex optimizer.
7	`MSK_IPAR_CHECK_CONVEXITY` Specify the level of convexity check on quadratic problems
70	`MSK_IPAR_LP_WRITE_IGNORE_INCOMPATIBLE_ITEMS` Controls the result of writing a problem containing incompatible items to an LP file.
134	`MSK_IPAR_READ_TASK_IGNORE_PARAM` Controls whether MOSEK should ignore the parameter setting defined in the task file and use the default parameter setting instead.
9	`MSK_IPAR_CHECK_TASK_DATA` If this feature is turned on, then the task data is checked for bad values i.e. NaNs. before an optimization is performed.
54	`MSK_IPAR_LOG_INTPNT` Controls amount of output printed printed by the interior-point optimizer. A higher level implies that more information is logged.
55	`MSK_IPAR_LOG_MIO` Controls the log level for the mixed integer optimizer. A higher level implies that more information is logged.
143	`MSK_IPAR_SIM_HOTSTART` Controls the type of hot-start that the simplex optimizer perform.
60	`MSK_IPAR_LOG_PARAM` Controls the amount of information printed out about parameter changes.
170	`MSK_IPAR_WRITE_DATA_FORMAT` Controls the file format when writing task data to a file.
73	`MSK_IPAR_MIO_BRANCH_DIR` Controls whether the mixed integer optimizer is branching up or down by default.
25	`MSK_IPAR_INTPNT_FACTOR_DEBUG_LVL` Controls factorization debug level.
18	`MSK_IPAR_FLUSH_STREAM_FREQ` Controls how frequent the message and log streams are flushed. A value of 0 means that it is never flushed. Otherwise a larger value results in less frequent flushes.

continued on next page

continued from previous page

161	MSK_IPAR_SOL_QUOTED_NAMES If this options is turned on, then MOSEK will quote names that contains blanks while writing the solution file. Moreover when reading leading and trailing quotes will be stripped of.
41	MSK_IPAR_LICENSE_PAUSE_TIME If MSK_IPAR_LICENSE_WAIT=MSK_ON and no license is available, then MOSEK sleeps a number of micro seconds between each check of whether a license as become free.
89	MSK_IPAR_MIO_PRESOLVE_AGGREGATE Controls whether the presolve used by the mixed integer optimizer tries to aggregate the constraints.
190	MSK_IPAR_WRITE_TASK_INC_SOL Controls whether the solutions are stored in the task file too.
38	MSK_IPAR_LICENSE_CACHE_TIME Controls the amount of time a license is cached in the MOSEK environment for reuse. Checking out a license from the license server has a small overhead. Therefore, if a large number of optimizations is performed within a small amount of time, it is efficient to cache the license in the MOSEK environment for later use. This way a number of license check outs from the license server is avoided. If a license has not been used in the given amount of time,MOSEK will automatically check in the license. To disable license caching set the value to 0.
4	MSK_IPAR_BI_MAX_ITERATIONS Controls the maximum number of simplex iterations allowed to optimize a basis after the basis identification.
103	MSK_IPAR_OPF_WRITE_SOLUTIONS Enable inclusion of solutions in the OPF files.
147	MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART This parameter controls has large the network component in “relative” terms has to be before it is exploited in a simplex hot-start. The network component should be equal or larger than max(MSK_IPAR_SIM_NETWORK_DETECT,MSK_IPAR_SIM_NETWORK_DETECT_HOTSTART) before it is exploited. If this value is larger than 100 the network flow component is never detected or exploited.
110	MSK_IPAR_PRESOLVE_LINDEP_USE

continued on next page

continued from previous page

	Controls whether the linear constraints are checked for linear dependencies.
106	`MSK_IPAR_PARAM_READ_IGN_ERROR` If turned on, then errors in paramter settings is ignored.
96	`MSK_IPAR_OPF_WRITE_HEADER` Write a text header with date and MOSEK version in an OPF file.
76	`MSK_IPAR_MIO_CONT_SOL` Controls the meaning of the interior-point and basic solutions in MIP problems.
94	`MSK_IPAR_OBJECTIVE_SENSE` If the objective sense for the task is undefined, then the value of this parameter is used as the default objective sense.
176	`MSK_IPAR_WRITE_INT_HEAD` Controls whether the header section is written to the integer solution file.
36	`MSK_IPAR_INTPNT_STARTING_POINT` Starting point used by the interior-point optimizer.
44	`MSK_IPAR_LOG` Controls the amount of log information. The value 0 implies that all log information is suppressed. A higher level implies that more information is logged. Please note that if a task is employed to solve a sequence of optimization problems the value of this parameter is reduced by the value of `MSK_IPAR_LOG_CUT_SECOND_OPT` for the second and any subsequent optimizations.
14	`MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX` Priority of the primal simplex algorithm when selecting solvers for concurrent optimization.
128	`MSK_IPAR_READ_MPS_OBJ_SENSE` If turned on, the MPS reader uses the objective sense section. Otherwise the MPS reader ignores it.
68	`MSK_IPAR_LOG_SIM_NETWORK_FREQ` Controls how frequent the network simplex optimizer outputs information about the optimization and how frequent the user-defined call-back function is called. The network optimizer will use a logging frequency equal to `MSK_IPAR_LOG_SIM_FREQ` times `MSK_IPAR_LOG_SIM_NETWORK_FREQ`.
24	`MSK_IPAR_INTPNT_DIFF_STEP`

continued on next page

continued from previous page

	Controls whether different step sizes are allowed in the primal and dual space.
155	`MSK_IPAR_SIM_SCALING` Controls how the problem is scaled before a simplex optimizer is used.
181	`MSK_IPAR_WRITE_LP_TERMS_PER_LINE` Maximum number of terms on a single line in an LP file written by MOSEK. 0 means unlimited.
136	`MSK_IPAR_SENSITIVITY_ALL` Not applicable.
160	`MSK_IPAR_SOL_FILTER_KEEP_RANGED` If turned on, then ranged constraints and variables are written to the solution file independent of the filter setting.
2	`MSK_IPAR_BI_IGNORE_MAX_ITER` If the parameter `MSK_IPAR_INTPNT_BASIS` has the value `MSK_BI_NO_ERROR` and the interior-point optimizer has terminated due to maximum number of iterations, then basis identification is performed if this parameter has the value `MSK_ON`.
50	`MSK_IPAR_LOG_FEASREPAIR` Controls the amount of output printed when performing feasibility repair.
35	`MSK_IPAR_INTPNT_SOLVE_FORM` Controls whether the primal or the dual problem is solved.
95	`MSK_IPAR_OPF_MAX_TERMS_PER_LINE` The maximum number of terms (linear and quadratic) per line when an OPF file is written.
186	`MSK_IPAR_WRITE_PRECISION` Controls the precision with which `double` numbers are printed in the MPS data file. In general it is not worthwhile to use a value higher than 15.
191	`MSK_IPAR_WRITE_XML_MODE` Controls if linear coefficients should be written by row or column when writing in the XML file format.
33	`MSK_IPAR_INTPNT_REGULARIZATION_USE` Controls whether regularization is allowed.
1	`MSK_IPAR_BI_CLEAN_OPTIMIZER` Controls which simplex optimizer is used in the clean-up phase.
192	`MSK_IPAR_MIO_PRESOLVE_PROBING`

continued on next page

continued from previous page

	Controls whether the mixed integer presolve performs probing. Probing can be very time consuming.
37	`MSK_IPAR_LICENSE_ALLOW_OVERUSE` Controls if license overuse is allowed when caching licenses
20	`MSK_IPAR_INFEAS_PREFER_PRIMAL` If both certificates of primal and dual infeasibility are supplied then only the primal is used when this option is turned on.
168	`MSK_IPAR_WRITE_BAS_VARIABLES` Controls whether the variables section is written to the basic solution file.
69	`MSK_IPAR_LOG_STORAGE` When turned on, MOSEK prints messages regarding the storage usage and allocation.
90	`MSK_IPAR_MIO_PRESOLVE_USE` Controls whether presolve is performed by the mixed integer optimizer.
111	`MSK_IPAR_PRESOLVE_LINDEP_WORK_LIM` Is used to limit the amount of work that can done to locate linear dependencies. In general the higher value this parameter is given the less work can be used. However, a value of 0 means no limit on the amount work that can be used.
23	`MSK_IPAR_INTPNT_BASIS` Controls whether the interior-point optimizer also computes an optimal basis.
48	`MSK_IPAR_LOG_CUT_SECOND_OPT` If a task is employed to solve a sequence of optimization problems, then the value of the log levels is reduced by the value of this parameter. E.g `MSK_IPAR_LOG` and `MSK_IPAR_LOG_SIM` are reduced by the value of this parameter for the second and any subsequent optimizations.
127	`MSK_IPAR_READ_MPS_KEEP_INT` Controls whether MOSEK should keep the integer restrictions on the variables while reading the MPS file.
85	`MSK_IPAR_MIO_MAX_NUM_SOLUTIONS` The mixed integer optimizer can be terminated after a certain number of different feasible solutions has been located. If this parameter has the value n and n is strictly positive, then the mixed integer optimizer will be terminated when n feasible solutions have been located.
39	`MSK_IPAR_LICENSE_CHECK_TIME`

continued on next page

continued from previous page

	The parameter specifies the number of seconds between the checks of all the active licenses in the MOSEK environment license cache. These checks are performed to determine if the licenses should be returned to the server.
189	`MSK_IPAR_WRITE_SOL_VARIABLES` Controls whether the variables section is written to the solution file.
137	`MSK_IPAR_SENSITIVITY_OPTIMIZER` Controls which optimizer is used for optimal partition sensitivity analysis.
182	`MSK_IPAR_WRITE_MPS_INT` Controls if marker records are written to the MPS file to indicate whether variables are integer restricted.
145	`MSK_IPAR_SIM_MAX_NUM_SETBACKS` Controls how many setbacks are allowed within a simplex optimizer. A setback is an event where the optimizer moves in the wrong direction. This is impossible in theory but may happen due to numerical problems.
16	`MSK_IPAR_DATA_CHECK` If this option is turned on, then extensive data checking is enabled. It will slow down MOSEK but on the other hand help locating bugs.
12	`MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX` Priority of the free simplex optimizer when selecting solvers for concurrent optimization.
123	`MSK_IPAR_READ_KEEP_FREE_CON` Controls whether the free constraints are included in the problem.
51	`MSK_IPAR_LOG_FILE` If turned on, then some log info is printed when a file is written or read.
13	`MSK_IPAR_CONCURRENT_PRIORITY_INTPNT` Priority of the interior-point algorithm when selecting solvers for concurrent optimization.
149	`MSK_IPAR_SIM_NON_SINGULAR` Controls if the simplex optimizer ensures a non-singular basis, if possible.
171	`MSK_IPAR_WRITE_DATA_PARAM` If this option is turned on the parameter settings are written to the data file as parameters.
139	`MSK_IPAR_SIM_DEGEN` Controls how aggressive degeneration is approached.

continued on next page

continued from previous page

97	`MSK_IPAR_OPF_WRITE_HINTS` Write a hint section with problem dimensions in the beginning of an OPF file.
108	`MSK_IPAR_PRESOLVE_ELIMINATOR_USE` Controls whether free or implied free variables are eliminated from the problem.
0	`MSK_IPAR_ALLOC_ADD_QNZ` Additional number of Q non-zeros that are allocated space for when `numanz` exceeds `maxnumqnz` during addition of new Q entries.
126	`MSK_IPAR_READ_MPS_FORMAT` Controls how strictly the MPS file reader interprets the MPS format.
105	`MSK_IPAR_PARAM_READ_CASE_NAME` If turned on, then names in the parameter file are case sensitive.
129	`MSK_IPAR_READ_MPS_QUOTED_NAMES` If a name is in quotes when reading an MPS file, then the quotes will be removed.
11	`MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX` Priority of the dual simplex algorithm when selecting solvers for concurrent optimization.
183	`MSK_IPAR_WRITE_MPS_OBJ_SENSE` If turned off, the objective sense section is not written to the MPS file.
30	`MSK_IPAR_INTPNT_NUM_THREADS` Controls the number of threads employed by the interior-point optimizer.
83	`MSK_IPAR_MIO_MAX_NUM_BRANCHES` Maximum number of branches allowed during the branch and bound search. A negative value means infinite.
150	`MSK_IPAR_SIM_PRIMAL_CRASH` Controls whether crashing is performed in the primal simplex optimizer. In general, if a basis consists of more than (100-this parameter value)% fixed variables, then a crash will be performed.
75	`MSK_IPAR_MIO_CONSTRUCT_SOL` If set to `MSK_ON` and all integer variables have been given a value for which a feasible MIP solution exists, then MOSEK generates an initial solution to the MIP by fixing all integer values and solving for the continuous variables.
92	`MSK_IPAR_MIO_STRONG_BRANCH`

continued on next page

continued from previous page

	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.
151	`MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION` The primal simplex optimizer can use a so-called restricted selection/pricing strategy to chooses the outgoing variable. Hence, if restricted selection is applied, then the primal simplex optimizer first choose a subset of all the potential incoming variables. Next, for some time it will choose the incoming variable only among the subset. From time to time the subset is redefined. A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.
120	`MSK_IPAR_READ_CONE` Expected maximum number of conic constraints to be read. The option is used only by fast MPS and LP file readers.
104	`MSK_IPAR_OPTIMIZER` Controls which optimizer is used to optimize the task.
71	`MSK_IPAR_MAX_NUM_WARNINGS` Waning level. A higher value results in more warnings.
42	`MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS` Controls whether license features expire warnings are suppressed.
188	`MSK_IPAR_WRITE_SOL_HEAD` Controls whether the header section is written to the solution file.
166	`MSK_IPAR_WRITE_BAS_CONSTRAINTS` Controls whether the constraint section is written to the basic solution file.
79	`MSK_IPAR_MIO_FEASPUMP_LEVEL` 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.
19	`MSK_IPAR_INFEAS_GENERIC_NAMES` Controls whether generic names are used when an infeasible subproblem is created.
146	`MSK_IPAR_SIM_NETWORK_DETECT`

continued on next page

continued from previous page

	The simplex optimizer is capable of exploiting a network flow component in a problem. However it is only worthwhile to exploit the network flow component if it is sufficiently large. This parameter controls how large the network component has to be in “relative” terms before it is exploited. For instance a value of 20 means at least 20% of the model should be a network before it is exploited. If this value is larger than 100 the network flow component is never detected or exploited.
62	`MSK_IPAR_LOG_RESPONSE` Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.
22	`MSK_IPAR_INFEAS_REPORT_LEVEL` Controls the amount of information presented in an infeasibility report. Higher values imply more information.
5	`MSK_IPAR_CACHE_SIZE_L1` Specifies the size of the cache of the computer. This parameter is potentially very important for the efficiency on computers if MOSEK cannot determine the cache size automatically. If the cache size is negative, then MOSEK tries to determine the value automatically.
6	`MSK_IPAR_CACHE_SIZE_L2` Specifies the size of the cache of the computer. This parameter is potentially very important for the efficiency on computers where MOSEK cannot determine the cache size automatically. If the cache size is negative, then MOSEK tries to determine the value automatically.
158	`MSK_IPAR_SIM_SWITCH_OPTIMIZER` The simplex optimizer sometimes chooses to solve the dual problem instead of the primal problem. This implies that if you have chosen to use the dual simplex optimizer and the problem is dualized, then it actually makes sense to use the primal simplex optimizer instead. If this parameter is on and the problem is dualized and furthermore the simplex optimizer is chosen to be the primal (dual) one, then it is switched to the dual (primal).
121	`MSK_IPAR_READ_DATA_COMPRESSED` If this option is turned on,it is assumed that the data file is compressed.
3	`MSK_IPAR_BI_IGNORE_NUM_ERROR`

continued on next page

continued from previous page

	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`.
99	`MSK_IPAR_OPF_WRITE_PROBLEM` Write objective, constraints, bounds etc. to an OPF file.
122	`MSK_IPAR_READ_DATA_FORMAT` Format of the data file to be read.
140	`MSK_IPAR_SIM_DUAL_CRASH` Controls whether crashing is performed in the dual simplex optimizer. In general if a basis consists of more than (100-this parameter value)% fixed variables, then a crash will be performed.
148	`MSK_IPAR_SIM_NETWORK_DETECT_METHOD` Controls which type of detection method the network extraction should use.
135	`MSK_IPAR_READ_VAR` Expected maximum number of variable to be read. The option is used only by MPS and LP file readers.
52	`MSK_IPAR_LOG_HEAD` If turned on, then a header line is added to the log.
154	`MSK_IPAR_SIM_SAVE_LU` Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.
26	`MSK_IPAR_INTPNT_FACTOR_METHOD` Controls the method used to factor the Newton equation system.
84	`MSK_IPAR_MIO_MAX_NUM_RELAXS` Maximum number of relaxations allowed during the branch and bound search. A negative value means infinite.
159	`MSK_IPAR_SOL_FILTER_KEEP_BASIC` If turned on, then basic and super basic constraints and variables are written to the solution file independent of the filter setting.
80	`MSK_IPAR_MIO_HEURISTIC_LEVEL` Controls the heuristic employed by the mixed integer optimizer to locate an initial good integer feasible solution. A value of zero means the heuristic is not used at all. A larger value than 0 means that a gradually more sophisticated heuristic is used which is computationally more expensive. A negative value implies that the optimizer chooses the heuristic. Normally a value around 3 to 5 should be optimal.

continued on next page

continued from previous page	
81	`MSK_IPAR_MIO_KEEP_BASIS` Controls whether the integer presolve keeps bases in memory. This speeds on the solution process at cost of bigger memory consumption.
34	`MSK_IPAR_INTPNT_SCALING` Controls how the problem is scaled before the interior-point optimizer is used.

D.19 Bound keys

Value	Name Description
0	`MSK_MARK_LO` The lower bound is selected for sensitivity analysis.
1	`MSK_MARK_UP` The upper bound is selected for sensitivity analysis.

D.20 Continuous mixed integer solution type

Value	Name Description
2	`MSK_MIO_CONT_SOL_ITG` The reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. A solution is only reported in case the problem has a primal feasible solution.
0	`MSK_MIO_CONT_SOL_NONE` No interior-point or basic solution are reported when the mixed integer optimizer is used.
1	`MSK_MIO_CONT_SOL_ROOT` The reported interior-point and basic solutions are a solution to the root node problem when mixed integer optimizer is used.
3	`MSK_MIO_CONT_SOL_ITG_REL` In case the problem is primal feasible then the reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. If the problem is primal infeasible, then the solution to the root node problem is reported.

D.21 Integer restrictions

Value	Name Description
0	`MSK_MIO_MODE_IGNORED` The integer constraints are ignored and the problem is solved as a continuous problem.
2	`MSK_MIO_MODE_LAZY` Integer restrictions should be satisfied if an optimizer is available for the problem.
1	`MSK_MIO_MODE_SATISFIED` Integer restrictions should be satisfied.

D.22 Mixed integer node selection types

Value	Name Description
5	`MSK_MIO_NODE_SELECTION_PSEUDO` The optimizer employs selects the node based on a pseudo cost estimate.
4	`MSK_MIO_NODE_SELECTION_HYBRID` The optimizer employs a hybrid strategy.
0	`MSK_MIO_NODE_SELECTION_FREE` The optimizer decides the node selection strategy.
3	`MSK_MIO_NODE_SELECTION_WORST` The optimizer employs a worst bound node selection strategy.
2	`MSK_MIO_NODE_SELECTION_BEST` The optimizer employs a best bound node selection strategy.
1	`MSK_MIO_NODE_SELECTION_FIRST` The optimizer employs a depth first node selection strategy.

D.23 MPS file format type

Value	Name Description
0	`MSK_MPS_FORMAT_STRICT` It is assumed that the input file satisfies the MPS format strictly.

continued on next page

continued from previous page

1	`MSK_MPS_FORMAT_RELAXED` It is assumed that the input file satisfies a slightly relaxed version of the MPS format.
2	`MSK_MPS_FORMAT_FREE` It is assumed that the input file satisfies the free MPS format. This implies that spaces are not allowed in names. Otherwise the format is free.

D.24 Message keys

Value	Name Description
1000	`MSK_MSG_READING_FILE` None
1001	`MSK_MSG_WRITING_FILE` None
1100	`MSK_MSG_MPS_SELECTED` None

D.25 Network detection method

Value	Name Description
1	`MSK_NETWORK_DETECT_SIMPLE` The network detection should use a very simple heuristic.
2	`MSK_NETWORK_DETECT_ADVANCED` The network detection should use a more advanced heuristic.
0	`MSK_NETWORK_DETECT_FREE` The network detection is free.

D.26 Objective sense types

Value	Name Description
1	`MSK_OBJECTIVE_SENSE_MINIMIZE`

continued on next page

continued from previous page

	The problem should be minimized.
0	`MSK_OBJECTIVE_SENSE_UNDEFINED` The objective sense is undefined.
2	`MSK_OBJECTIVE_SENSE_MAXIMIZE` The problem should be maximized.

D.27 On/off

Value	Name Description
1	`MSK_ON` Switch the option on.
0	`MSK_OFF` Switch the option off.

D.28 Optimizer types

Value	Name Description
1	`MSK_OPTIMIZER_INTPNT` The interior-point optimizer is used.
9	`MSK_OPTIMIZER_CONCURRENT` The optimizer for nonconvex nonlinear problems.
7	`MSK_OPTIMIZER_MIXED_INT` The mixed integer optimizer.
5	`MSK_OPTIMIZER_DUAL_SIMPLEX` The dual simplex optimizer is used.
0	`MSK_OPTIMIZER_FREE` The optimizer is chosen automatically.
2	`MSK_OPTIMIZER_CONIC` Another cone optimizer.
8	`MSK_OPTIMIZER_NONCONVEX` The optimizer for nonconvex nonlinear problems.
3	`MSK_OPTIMIZER_QCONE` The Qcone optimizer is used.
4	`MSK_OPTIMIZER_PRIMAL_SIMPLEX`

continued on next page

continued from previous page

	The primal simplex optimizer is used.
6	`MSK_OPTIMIZER_FREE_SIMPLEX` Either the primal or the dual simplex optimizer is used.

D.29 Ordering strategies

Value	Name Description
5	`MSK_ORDER_METHOD_NONE` No ordering is used.
2	`MSK_ORDER_METHOD_APPMINLOC2` A variant of the approximate minimum local-fill-in ordering is used.
1	`MSK_ORDER_METHOD_APPMINLOC1` Approximate minimum local-fill-in ordering is used.
4	`MSK_ORDER_METHOD_GRAPHPAR2` An alternative graph partitioning based ordering.
0	`MSK_ORDER_METHOD_FREE` The ordering method is chosen automatically.
3	`MSK_ORDER_METHOD_GRAPHPAR1` Graph partitioning based ordering.

D.30 Parameter type

Value	Name Description
0	`MSK_PAR_INVALID_TYPE` Not a valid parameter.
3	`MSK_PAR_STR_TYPE` Is a string parameter.
1	`MSK_PAR_DOU_TYPE` Is a double parameter.
2	`MSK_PAR_INT_TYPE` Is an integer parameter.

D.31 Presolve method.

Value	Name Description
1	`MSK_PRESOLVE_MODE_ON` The problem is presolved before it is optimized.
0	`MSK_PRESOLVE_MODE_OFF` The problem is not presolved before it is optimized.
2	`MSK_PRESOLVE_MODE_FREE` It is decided automatically whether to presolve before the problem is optimized.

D.32 Problem data items

Value	Name Description
0	`MSK_PI_VAR` Item is a variable.
2	`MSK_PI_CONE` Item is a cone.
1	`MSK_PI_CON` Item is a constraint.

D.33 Problem types

Value	Name Description
2	`MSK_PROBTYPE_QCQO` The problem is a quadratically constrained optimization problem.
0	`MSK_PROBTYPE_LO` The problem is a linear optimization problem.
4	`MSK_PROBTYPE_CONIC` A conic optimization.
3	`MSK_PROBTYPE_GECO` General convex optimization.
5	`MSK_PROBTYPE_MIXED` General nonlinear constraints and conic constraints. This combination can not be solved by MOSEK.
1	`MSK_PROBTYPE_QO` The problem is a quadratic optimization problem.

D.34 Problem status keys

Value	Name Description
6	`MSK_PRO_STA_PRIM_AND_DUAL_INFEAS` The problem is primal and dual infeasible.
4	`MSK_PRO_STA_PRIM_INFEAS` The problem is primal infeasible.
7	`MSK_PRO_STA_ILL_POSED` The problem is ill-posed. For example, it may be primal and dual feasible but have a positive duality gap.
0	`MSK_PRO_STA_UNKNOWN` Unknown problem status.
2	`MSK_PRO_STA_PRIM_FEAS` The problem is primal feasible.
8	`MSK_PRO_STA_NEAR_PRIM_AND_DUAL_FEAS` The problem is at least nearly primal and dual feasible.
10	`MSK_PRO_STA_NEAR_DUAL_FEAS` The problem is at least nearly dual feasible.
11	`MSK_PRO_STA_PRIM_INFEAS_OR_UNBOUNDED` The problem is either primal infeasible or unbounded. This may occur for mixed integer problems.
1	`MSK_PRO_STA_PRIM_AND_DUAL_FEAS` The problem is primal and dual feasible.
5	`MSK_PRO_STA_DUAL_INFEAS` The problem is dual infeasible.
9	`MSK_PRO_STA_NEAR_PRIM_FEAS` The problem is at least nearly primal feasible.
3	`MSK_PRO_STA_DUAL_FEAS` The problem is dual feasible.

D.35 Interpretation of quadratic terms in MPS files

Value	Name Description
0	`MSK_Q_READ_ADD` All elements in a Q matrix are assumed to belong to the lower triangular part. Duplicate elements in a Q matrix are added together.

continued on next page

continued from previous page

1	`MSK_Q_READ_DROP_LOWER`	All elements in the strict lower triangular part of the Q matrices are dropped.
2	`MSK_Q_READ_DROP_UPPER`	All elements in the strict upper triangular part of the Q matrices are dropped.

D.36 Response codes

Value	Name	Description
1218	`MSK_RES_ERR_PARAM_TYPE`	The parameter type is invalid.
1268	`MSK_RES_ERR_INV_SKX`	Invalid value in `skx`.
1203	`MSK_RES_ERR_INDEX_IS_TOO_SMALL`	An index in an argument is too small.
803	`MSK_RES_WRN_PRESOLVE_BAD_PRECISION`	The presolve estimates that the model is specified with insufficient precision.
1500	`MSK_RES_ERR_INV_PROBLEM`	Invalid problem type. Probably a nonconvex problem has been specified.
2505	`MSK_RES_ERR_CANNOT_CLONE_NL`	A task with a nonlinear function call-back cannot be cloned.
1551	`MSK_RES_ERR_MIO_NO_OPTIMIZER`	No optimizer is available for the current class of integer optimization problems.
3003	`MSK_RES_ERR_API_NL_DATA`	None
4004	`MSK_RES_TRM_MIO_NEAR_ABS_GAP`	The mixed-integer optimizer terminated because the near optimal absolute gap tolerance was satisfied.
1254	`MSK_RES_ERR_MUL_A_ELEMENT`	An element in A is defined multiple times.
1170	`MSK_RES_ERR_INVALID_NAME_IN_SOL_FILE`	An invalid name occurred in a solution file.
1114	`MSK_RES_ERR_MPS_MUL_QOBJ`	

continued on next page

continued from previous page

	The Q term in the objective is specified multiple times in the MPS data file.
1063	`MSK_RES_ERR_NO_INIT_ENV` `env` is not initialized.
1265	`MSK_RES_ERR_UNDEF_SOLUTION` The required solution is not defined.
1288	`MSK_RES_ERR_LASTJ` Invalid `lastj`.
1001	`MSK_RES_ERR_LICENSE_EXPIRED` The license has expired.
3055	`MSK_RES_ERR_SEN_INDEX_INVALID` Invalid range given in the sensitivity file.
1274	`MSK_RES_ERR_INV_SKN` Invalid value in `skn`.
1295	`MSK_RES_ERR_OBJ_Q_NOT_PSD` The quadratic coefficient matrix in the objective is not positive semidefinite as expected for a minimization problem.
1267	`MSK_RES_ERR_INV_SKC` Invalid value in `skc`.
1008	`MSK_RES_ERR_MISSING_LICENSE_FILE` MOSEK cannot find the license file or license server. Usually this happens if the operating system variable `MOSEKLM_LICENSE_FILE` is not set up appropriately. Please see the MOSEK installation manual for details.
1235	`MSK_RES_ERR_INDEX` An index is out of range.
3058	`MSK_RES_ERR_SEN_NUMERICAL` Numerical difficulties encountered performing the sensitivity analysis.
2800	`MSK_RES_ERR_LU_MAX_NUM_TRIES` Could not compute the LU factors of the matrix within the maximum number of allowed tries.
201	`MSK_RES_WRN_DROPPED_NZ_QOBJ` One or more non-zero elements were dropped in the Q matrix in the objective.
3000	`MSK_RES_ERR_INTERNAL` An internal error occurred. Please report this problem.
800	`MSK_RES_WRN_NONCOMPLETE_LINEAR_DEPENDENCY_CHECK` The linear dependency check(s) was not completed and therefore the A matrix may contain linear dependencies.

continued on next page

continued from previous page

1204	`MSK_RES_ERR_INDEX_IS_TOO_LARGE` An index in an argument is too large.
1154	`MSK_RES_ERR_LP_INVALID_VAR_NAME` A variable name is invalid when used in an LP formatted file.
2950	`MSK_RES_ERR_NO_DUAL_FOR_ITG_SOL` No dual information is available for the integer solution.
1150	`MSK_RES_ERR_LP_INCOMPATIBLE` The problem cannot be written to an LP formatted file.
1501	`MSK_RES_ERR_MIXED_PROBLEM` The problem contains both conic and nonlinear constraints.
1700	`MSK_RES_ERR_FEASREPAIR_CANNOT_RELAX` An optimization problem cannot be relaxed. This is the case e.g. for general nonlinear optimization problems.
1207	`MSK_RES_ERR_PARAM_NAME_INT` The parameter name is not correct for an integer parameter.
3057	`MSK_RES_ERR_SEN_SOLUTION_STATUS` No optimal solution found to the original problem given for sensitivity analysis.
1432	`MSK_RES_ERR_USER_NLO_FUNC` The user-defined nonlinear function reported an error.
405	`MSK_RES_WRN_TOO_MANY_BASIS_VARS` A basis with too many variables has been specified.
1081	`MSK_RES_ERR_SPACE_NO_INFO` No available information about the space usage.
1205	`MSK_RES_ERR_PARAM_NAME` The parameter name is not correct.
3056	`MSK_RES_ERR_SEN_INVALID_REGEXP` Syntax error in regexp or regexp longer than 1024.
200	`MSK_RES_WRN_NZ_IN_UPR_TRI` Non-zero elements specified in the upper triangle of a matrix were ignored.
505	`MSK_RES_WRN_LICENSE_FEATURE_EXPIRE` The license expires.
1263	`MSK_RES_ERR_NEGATIVE_SURPLUS` Negative surplus.
1404	`MSK_RES_ERR_INV_QCON_SUBK` Invalid value in `qcsubk`.
1406	`MSK_RES_ERR_INV_QCON_SUBJ` Invalid value in `qcsubj`.

continued on next page

continued from previous page

1198	`MSK_RES_ERR_ARGUMENT_TYPE` Incorrect argument type.
1017	`MSK_RES_ERR_LICENSE_MOSEKLM_DAEMON` The MOSEKLM license manager daemon is not up and running.
2901	`MSK_RES_ERR_INVALID_WCHAR` An invalid `wchar` string is encountered.
1059	`MSK_RES_ERR_END_OF_FILE` End of file reached.
1462	`MSK_RES_ERR_NAN_IN_BUC` u^c contains an invalid floating point value, i.e. a `NaN`.
1290	`MSK_RES_ERR_NONLINEAR_EQUALITY` The model contains a nonlinear equality which defines a nonconvex set.
1055	`MSK_RES_ERR_DATA_FILE_EXT` The data file format cannot be determined from the file name.
1210	`MSK_RES_ERR_PARAM_INDEX` Parameter index is out of range.
1285	`MSK_RES_ERR_FIRSTI` Invalid `firsti`.
1000	`MSK_RES_ERR_LICENSE` Invalid license.
1299	`MSK_RES_ERR_ARGUMENT_PERM_ARRAY` An invalid permutation array is specified.
85	`MSK_RES_WRN_LP_DROP_VARIABLE` Ignored a variable because the variable was not previously defined. Usually this implies that a variable appears in the bound section but not in the objective or the constraints.
1287	`MSK_RES_ERR_FIRSTJ` Invalid `firstj`.
1090	`MSK_RES_ERR_READ_FORMAT` The specified format cannot be read.
1219	`MSK_RES_ERR_INF_DOU_INDEX` A double information index is out of range for the specified type.
1286	`MSK_RES_ERR_LASTI` Invalid `lasti`.
1199	`MSK_RES_ERR_NR_ARGUMENTS` Incorrect number of function arguments.
1293	`MSK_RES_ERR_CON_Q_NOT_PSD`

continued on next page

continued from previous page

		The quadratic constraint matrix is not positive semi-definite as expected for a constraint with finite upper bound. This results in a nonconvex problem.
63	MSK_RES_WRN_ZERO_AIJ	One or more zero elements are specified in A.
2504	MSK_RES_ERR_INV_NUMJ	Invalid numj.
1650	MSK_RES_ERR_FACTOR	An error occurred while factorizing a matrix.
3201	MSK_RES_ERR_INVALID_BRANCH_PRIORITY	An invalid branching priority is specified. It should be nonnegative.
1216	MSK_RES_ERR_PARAM_IS_TOO_SMALL	The parameter value is too small.
1163	MSK_RES_ERR_LP_WRITE_CONIC_PROBLEM	The problem contains cones that cannot be written to an LP formatted file.
4000	MSK_RES_TRM_MAX_ITERATIONS	The optimizer terminated at the maximum number of iterations.
1240	MSK_RES_ERR_MAXNUMCON	The maximum number of constraints specified is smaller than the number of constraints in the task.
1050	MSK_RES_ERR_UNKNOWN	Unknown error.
1162	MSK_RES_ERR_READ_LP_NONEXISTING_NAME	A variable never occurred in objective or constraints.
2503	MSK_RES_ERR_INV_NUMI	Invalid numi.
1292	MSK_RES_ERR_NONLINEAR_RANGED	The model contains a nonlinear ranged constraint which by definition defines a nonconvex set.
1047	MSK_RES_ERR_THREAD_MUTEX_UNLOCK	Could not unlock a mutex.
1100	MSK_RES_ERR_MPS_FILE	An error occurred while reading an MPS file.
1156	MSK_RES_ERR_WRITE_OPF_INVALID_VAR_NAME	Empty variable names cannot be written to OPF files.
1152	MSK_RES_ERR_LP_DUP_SLACK_NAME	The name of the slack variable added to a ranged constraint already exists.

continued on next page

continued from previous page

2000	`MSK_RES_ERR_NO_PRIMAL_INFEAS_CER` A certificate of primal infeasibility is not available.
1158	`MSK_RES_ERR_WRITE_LP_FORMAT` Problem cannot be written as an LP file.
3052	`MSK_RES_ERR_SEN_INDEX_RANGE` Index out of range in the sensitivity analysis file.
66	`MSK_RES_WRN_SPAR_MAX_LEN` A value for a string parameter is longer than the buffer that is supposed to hold it.
3050	`MSK_RES_ERR_SEN_FORMAT` Syntax error in sensitivity analysis file.
1407	`MSK_RES_ERR_INV_QCON_VAL` Invalid value in `qcval`.
1206	`MSK_RES_ERR_PARAM_NAME_DOU` The parameter name is not correct for a double parameter.
1291	`MSK_RES_ERR_NONCONVEX` The optimization problem is nonconvex.
1300	`MSK_RES_ERR_CONE_INDEX` An index of a non-existing cone has been specified.
1470	`MSK_RES_ERR_NAN_IN_C` c contains an invalid floating point value, i.e. a `NaN`.
1304	`MSK_RES_ERR_MAXNUMCONE` The value specified for `maxnumcone` is too small.
1103	`MSK_RES_ERR_MPS_NULL_CON_NAME` An empty constraint name is used in an MPS file.
1417	`MSK_RES_ERR_QCON_UPPER_TRIANGLE` An element in the upper triangle of a Q^k is specified. Only elements in the lower triangle should be specified.
4015	`MSK_RES_TRM_NUM_MAX_NUM_INT_SOLUTIONS` The mixed-integer optimizer terminated as the maximum number of feasible solutions was reached.
1125	`MSK_RES_ERR_MPS_TAB_IN_FIELD2` A tab char occurred in field 2.
270	`MSK_RES_WRN_MIO_INFEASIBLE_FINAL` The final mixed integer problem with all the integer variables fixed at their optimal values is infeasible.
1251	`MSK_RES_ERR_NUMVARLIM` Maximum number of variables limit is exceeded.
1433	`MSK_RES_ERR_USER_NLO_EVAL`

continued on next page

continued from previous page

		The user-defined nonlinear function reported an error.
1232	`MSK_RES_ERR_INF_TYPE`	The information type is invalid.
1106	`MSK_RES_ERR_MPS_UNDEF_VAR_NAME`	An undefined variable name occurred in an MPS file.
503	`MSK_RES_WRN_USING_GENERIC_NAMES`	The file writer reverts to generic names because a name is blank.
1127	`MSK_RES_ERR_MPS_TAB_IN_FIELD5`	A tab char occurred in field 5.
1056	`MSK_RES_ERR_INVALID_FILE_NAME`	An invalid file name has been specified.
804	`MSK_RES_WRN_WRITE_DISCARDED_CFIX`	The fixed objective term could not be converted to a variable and was discarded in the output file.
1415	`MSK_RES_ERR_QOBJ_UPPER_TRIANGLE`	An element in the upper triangle of Q^o is specified. Only elements in the lower triangle should be specified.
1054	`MSK_RES_ERR_FILE_WRITE`	File write error.
1164	`MSK_RES_ERR_LP_WRITE_GECO_PROBLEM`	The problem contains general convex terms that cannot be written to an LP formatted file.
1243	`MSK_RES_ERR_MAXNUMQNZ`	The maximum number of non-zeros specified for the Q matrices is smaller than the number of non-zeros in the current Q matrices.
2506	`MSK_RES_ERR_CANNOT_HANDLE_NL`	A function cannot handle a task with nonlinear function call-backs.
1600	`MSK_RES_ERR_NO_BASIS_SOL`	No basic solution is defined.
1131	`MSK_RES_ERR_ORD_INVALID`	Invalid content in branch ordering file.
1303	`MSK_RES_ERR_CONE_REP_VAR`	A variable is included multiple times in the cone.
1075	`MSK_RES_ERR_INVALID_OBJ_NAME`	An invalid objective name is specified.
1052	`MSK_RES_ERR_FILE_OPEN`	Error while opening a file.
250	`MSK_RES_WRN_IGNORE_INTEGER`	Ignored integer constraints.

continued on next page

continued from previous page

1296	`MSK_RES_ERR_OBJ_Q_NOT_NSD` The quadratic coefficient matrix in the objective is not negative semidefinite as expected for a maximization problem.
1064	`MSK_RES_ERR_INVALID_TASK` The `task` is invalid.
1065	`MSK_RES_ERR_NULL_POINTER` An argument to a function is unexpectedly a NULL pointer.
3005	`MSK_RES_ERR_API_FATAL_ERROR` An internal error occurred in the API. Please report this problem.
1550	`MSK_RES_ERR_INV_OPTIMIZER` An invalid optimizer has been chosen for the problem. This means that the simplex or the conic optimizer is chosen to optimize a nonlinear problem.
1310	`MSK_RES_ERR_REMOVE_CONE_VARIABLE` A variable cannot be removed because it will make a cone invalid.
62	`MSK_RES_WRN_LARGE_AIJ` A numerically large value is specified for one $a_{i,j}$.
1208	`MSK_RES_ERR_PARAM_NAME_STR` The parameter name is not correct for a string parameter.
1018	`MSK_RES_ERR_LICENSE_FEATURE` A requested feature is not available in the license file(s). Most likely due to an incorrect license system setup.
251	`MSK_RES_WRN_NO_GLOBAL_OPTIMIZER` No global optimizer is available.
1040	`MSK_RES_ERR_LINK_FILE_DLL` A file cannot be linked to a stream in the DLL version.
1701	`MSK_RES_ERR_FEASREPAIR_SOLVING_RELAXED` The relaxed problem could not be solved to optimality. Please consult the log file for further details.
1221	`MSK_RES_ERR_INDEX_ARR_IS_TOO_SMALL` An index in an array argument is too small.
1259	`MSK_RES_ERR_SOLVER_PROBTYPE` Problem type does not match the chosen optimizer.
1220	`MSK_RES_ERR_INF_INT_INDEX` An integer information index is out of range for the specified type.
1053	`MSK_RES_ERR_FILE_READ` File read error.
1440	`MSK_RES_ERR_USER_NLO_EVAL_HESSUBI`

continued on next page

continued from previous page

		The user-defined nonlinear function reported an invalid subscript in the Hessian.
1441	`MSK_RES_ERR_USER_NLO_EVAL_HESSUBJ`	The user-defined nonlinear function reported an invalid subscript in the Hessian.
300	`MSK_RES_WRN_SOL_FILTER`	Invalid solution filter is specified.
3100	`MSK_RES_ERR_UNB_STEP_SIZE`	A step size in an optimizer was unexpectedly unbounded. For instance, if the step-size becomes unbounded in phase 1 of the simplex algorithm then an error occurs. Normally this will happen only if the problem is badly formulated. Please contact MOSEK support if this error occurs.
4030	`MSK_RES_TRM_INTERNAL`	The optimizer terminated due to some internal reason. Please contact MOSEK support.
1110	`MSK_RES_ERR_MPS_NO_OBJECTIVE`	No objective is defined in an MPS file.
1400	`MSK_RES_ERR_INFINITE_BOUND`	A finite bound value is too large in absolute value.
1030	`MSK_RES_ERR_OPEN_DL`	A dynamic link library could not be opened.
3001	`MSK_RES_ERR_API_ARRAY_TOO_SMALL`	An input array was too short.
1046	`MSK_RES_ERR_THREAD_MUTEX_LOCK`	Could not lock a mutex.
1262	`MSK_RES_ERR_LAST`	Invalid `last`.
1151	`MSK_RES_ERR_LP_EMPTY`	The problem cannot be written to an LP formatted file.
1011	`MSK_RES_ERR_SIZE_LICENSE_VAR`	The problem has too many variables to be solved with the available license.
1062	`MSK_RES_ERR_INVALID_STREAM`	An invalid stream is referenced.
2520	`MSK_RES_ERR_INVALID_ACCMODE`	An invalid access mode is specified.
1250	`MSK_RES_ERR_NUMCONLIM`	Maximum number of constraints limit is exceeded.

continued on next page

continued from previous page

1104	`MSK_RES_ERR_MPS_NULL_VAR_NAME` An empty variable name is used in an MPS file.
72	`MSK_RES_WRN_MPS_SPLIT_BOU_VECTOR` A BOUNDS vector is split into several nonadjacent parts in an MPS file.
1026	`MSK_RES_ERR_LICENSE_SERVER_VERSION` The version specified in the checkout request is greater than the highest version number the daemon supports.
1025	`MSK_RES_ERR_LICENSE_INVALID_HOSTID` The host ID specified in the license file does not match the host ID of the computer.
1045	`MSK_RES_ERR_THREAD_MUTEX_INIT` Could not initialize a mutex.
1280	`MSK_RES_ERR_INV_NAME_ITEM` An invalid name item code is used.
53	`MSK_RES_WRN_LARGE_UP_BOUND` A large but finite upper bound in absolute value has been specified.
4009	`MSK_RES_TRM_MIO_NUM_BRANCHES` The mixed-integer optimizer terminated as to the maximum number of branches was reached.
1272	`MSK_RES_ERR_INV_CONE_TYPE` Invalid cone type code is encountered.
1112	`MSK_RES_ERR_MPS_MUL_CON_NAME` A constraint name was specified multiple times in the `ROWS` section.
1801	`MSK_RES_ERR_INVALID_IOMODE` Invalid io mode.
1115	`MSK_RES_ERR_MPS_INV_SEC_ORDER` The sections in the MPS data file are not in the correct order.
1016	`MSK_RES_ERR_LICENSE_MAX` Maximum number of licenses is reached.
4007	`MSK_RES_TRM_USER_CALLBACK` The optimizer terminated due to the return of the user-defined callback function.
1430	`MSK_RES_ERR_USER_FUNC_RET` An user function reported an error.
1058	`MSK_RES_ERR_INVALID_MBT_FILE` A MOSEK binary task file is invalid.
1294	`MSK_RES_ERR_CON_Q_NOT_NSD`

continued on next page

continued from previous page

	The quadratic constraint matrix is not negative semi-definite as expected for a constraint with finite lower bound. This results in a nonconvex problem.
3600	`MSK_RES_ERR_XML_INVALID_PROBLEM_TYPE` The problem type is not supported by the XML format.
1231	`MSK_RES_ERR_INF_INT_NAME` A integer information name is invalid.
1107	`MSK_RES_ERR_MPS_INV_CON_KEY` An invalid constraint key occurred in an MPS file.
2001	`MSK_RES_ERR_NO_DUAL_INFEAS_CER` A certificate of infeasibility is not available.
4025	`MSK_RES_TRM_NUMERICAL_PROBLEM` The optimizer terminated due to numerical problems.
52	`MSK_RES_WRN_LARGE_LO_BOUND` A large but finite lower bound in absolute value has been specified.
3999	`MSK_RES_ERR_API_INTERNAL` None
70	`MSK_RES_WRN_MPS_SPLIT_RHS_VECTOR` An RHS vector is split into several nonadjacent parts in an MPS file.
3053	`MSK_RES_ERR_SEN_BOUND_INVALID_UP` Analysis of upper bound requested for an index, where no upper bound exists.
1702	`MSK_RES_ERR_FEASREPAIR_INCONSISTENT_BOUND` The upper bound is less than the lower bound for a variable or a constraint. Please correct this before running the feasibility repair.
1449	`MSK_RES_ERR_Y_IS_UNDEFINED` The solution item y is undefined.
3200	`MSK_RES_ERR_INVALID_BRANCH_DIRECTION` An invalid branching direction is specified.
3004	`MSK_RES_ERR_API_CALLBACK` None
1305	`MSK_RES_ERR_CONE_TYPE` Invalid cone type specified.
4008	`MSK_RES_TRM_MIO_NUM_RELAXS` The mixed-integer optimizer terminated as the maximum number of relaxations was reached.
1256	`MSK_RES_ERR_INV_BKC` Invalid bound key is specified for a constraint.
4020	`MSK_RES_TRM_MAX_NUM_SETBACKS`

continued on next page

continued from previous page

	The optimizer terminated as the maximum number of set-backs was reached. This indicates numerical problems and a possibly badly formulated problem.
3101	`MSK_RES_ERR_IDENTICAL_TASKS` Some tasks related to this function call were identical. Unique tasks were expected.
1020	`MSK_RES_ERR_LICENSE_CANNOT_ALLOCATE` The license system cannot allocate the memory required.
1402	`MSK_RES_ERR_INV_QOBJ_SUBJ` Invalid value in `qosubj`.
1302	`MSK_RES_ERR_CONE_OVERLAP` A new cone which variables overlap with an existing cone has been specified.
1401	`MSK_RES_ERR_INV_QOBJ_SUBI` Invalid value in `qosubi`.
1153	`MSK_RES_ERR_WRITE_MPS_INVALID_NAME` An invalid name is created while writing an MPS file. Usually this will make the MPS file unreadable.
1553	`MSK_RES_ERR_MIO_NOT_LOADED` The mixed-integer optimizer is not loaded.
1061	`MSK_RES_ERR_NULL_TASK` `task` is a NULL pointer.
1450	`MSK_RES_ERR_NAN_IN_DOUBLE_DATA` An invalid floating point value was used in some double data.
3059	`MSK_RES_ERR_CONCURRENT_OPTIMIZER` An unsupported optimizer was chosen for use with the concurrent optimizer.
1252	`MSK_RES_ERR_TOO_SMALL_MAXNUMANZ` Maximum number of non-zeros allowed in A is too small.
1197	`MSK_RES_ERR_ARGUMENT_LENNEQ` Incorrect length of arguments.
500	`MSK_RES_WRN_LICENSE_EXPIRE` The license expires.
1200	`MSK_RES_ERR_IN_ARGUMENT` A function argument is incorrect.
1051	`MSK_RES_ERR_SPACE` Out of space.
1241	`MSK_RES_ERR_MAXNUMVAR`

continued on next page

continued from previous page

	The maximum number of variables specified is smaller than the number of variables in the task.
1800	`MSK_RES_ERR_INVALID_COMPRESSION` Invalid compression type.
1101	`MSK_RES_ERR_MPS_INV_FIELD` A field in the MPS file is invalid. Probably it is too wide.
1060	`MSK_RES_ERR_NULL_ENV` `env` is a NULL pointer.
3500	`MSK_RES_ERR_INTERNAL_TEST_FAILED` An internal unit test function failed.
501	`MSK_RES_WRN_LICENSE_SERVER` The license server is not responding.
1122	`MSK_RES_ERR_MPS_INVALID_OBJSENSE` An invalid objective sense is specified.
1168	`MSK_RES_ERR_OPF_FORMAT` Syntax error in an OPF file
1269	`MSK_RES_ERR_INV_SK_STR` Invalid status key string encountered.
1071	`MSK_RES_ERR_DUP_NAME` An error occurred while reading an MPS file..
1116	`MSK_RES_ERR_MPS_MUL_CSEC` Multiple `CSECTION`s are given the same name.
51	`MSK_RES_WRN_LARGE_BOUND` A very large bound in absolute value has been specified.
50	`MSK_RES_WRN_OPEN_PARAM_FILE` The parameter file could not be opened.
1431	`MSK_RES_ERR_USER_FUNC_RET_DATA` An user function returned invalid data.
1615	`MSK_RES_ERR_BASIS_SINGULAR` The basis is singular and hence cannot be factored.
1155	`MSK_RES_ERR_LP_FREE_CONSTRAINT` Free constraints cannot be written in LP file format.
1445	`MSK_RES_ERR_INVALID_OBJECTIVE_SENSE` An invalid objective sense is specified.
0	`MSK_RES_OK` No error occurred.
3002	`MSK_RES_ERR_API_CB_CONNECT` None
1253	`MSK_RES_ERR_INV_APTRE`

continued on next page

continued from previous page

	`aptre[j]` is strictly smaller than `aptrb[j]` for some `j`.
1013	`MSK_RES_ERR_OPTIMIZER_LICENSE` The optimizer required is not licensed.
1007	`MSK_RES_ERR_FILE_LICENSE` Invalid license file.
1160	`MSK_RES_ERR_LP_FORMAT` Syntax error in an LP file.
1237	`MSK_RES_ERR_SOLITEM` The solution item number `solitem` is invalid. Please note `MSK_SOL_ITEM_SNX` is invalid for the basic solution.
1010	`MSK_RES_ERR_SIZE_LICENSE_CON` The problem has too many constraints to be solved with the available license.
1118	`MSK_RES_ERR_MPS_CONE_OVERLAP` A variable is specified to be a member of several cones.
700	`MSK_RES_WRN_ZEROS_IN_SPARSE_DATA` One or more almost zero elements are specified in sparse input data.
1408	`MSK_RES_ERR_QCON_SUBI_TOO_SMALL` Invalid value in `qcsubi`.
1610	`MSK_RES_ERR_BASIS_FACTOR` The factorization of the basis is invalid.
1580	`MSK_RES_ERR_POSTSOLVE` An error occurred during the postsolve. Please contact MOSEK support.
1215	`MSK_RES_ERR_PARAM_IS_TOO_LARGE` The parameter value is too large.
1281	`MSK_RES_ERR_PRO_ITEM` An invalid problem is used.
1057	`MSK_RES_ERR_INVALID_SOL_FILE_NAME` An invalid file name has been specified.
1271	`MSK_RES_ERR_INV_CONE_TYPE_STR` Invalid cone type string encountered.
1283	`MSK_RES_ERR_INVALID_FORMAT_TYPE` Invalid format type.
57	`MSK_RES_WRN_LARGE_CJ` A numerically large value is specified for one c_j.
1035	`MSK_RES_ERR_OLDER_DLL` The dynamic link library is older than the specified version.
1019	`MSK_RES_ERR_PLATFORM_NOT_LICENSED`

continued on next page

continued from previous page

	A requested license feature is not available for the required platform.
1119	`MSK_RES_ERR_MPS_CONE_REPEAT` A variable is repeated within the `CSECTION`.
3051	`MSK_RES_ERR_SEN_UNDEF_NAME` An undefined name was encountered in the sensitivity analysis file.
1403	`MSK_RES_ERR_INV_QOBJ_VAL` Invalid value in `qoval`.
71	`MSK_RES_WRN_MPS_SPLIT_RAN_VECTOR` A RANGE vector is split into several nonadjacent parts in an MPS file.
3054	`MSK_RES_ERR_SEN_BOUND_INVALID_LO` Analysis of lower bound requested for an index, where no lower bound exists.
1111	`MSK_RES_ERR_MPS_SPLITTED_VAR` A variable is split in an MPS data file.
1080	`MSK_RES_ERR_SPACE_LEAKING` MOSEK is leaking memory. This can be due to either an incorrect use of MOSEK or a bug.
1201	`MSK_RES_ERR_ARGUMENT_DIMENSION` A function argument is of incorrect dimension.
280	`MSK_RES_WRN_FIXED_BOUND_VALUES` A fixed constraint/variable has been specified using the bound keys but the numerical bounds are different. The variable is fixed at the lower bound.
4001	`MSK_RES_TRM_MAX_TIME` The optimizer terminated at the maximum amount of time.
1461	`MSK_RES_ERR_NAN_IN_BLC` l^c contains an invalid floating point value, i.e. a `NaN`.
1350	`MSK_RES_ERR_SOL_FILE_NUMBER` An invalid number is specified in a solution file.
3700	`MSK_RES_ERR_INVALID_AMPL_STUB` Invalid AMPL stub.
1260	`MSK_RES_ERR_OBJECTIVE_RANGE` Empty objective range.
1238	`MSK_RES_ERR_WHICHITEM_NOT_ALLOWED` `whichitem` is unacceptable.
1471	`MSK_RES_ERR_NAN_IN_BLX` l^x contains an invalid floating point value, i.e. a `NaN`.
1236	`MSK_RES_ERR_WHICHSOL`

continued on next page

continued from previous page

	The solution defined by compwhichsol does not exists.
1242	`MSK_RES_ERR_MAXNUMANZ` The maximum number of non-zeros specified for A is smaller than the number of non-zeros in the current A.
801	`MSK_RES_WRN_ELIMINATOR_SPACE` The eliminator is skipped at least once due to lack of space.
1049	`MSK_RES_ERR_THREAD_COND_INIT` Could not initialize a condition.
1405	`MSK_RES_ERR_INV_QCON_SUBI` Invalid value in `qcsubi`.
1036	`MSK_RES_ERR_NEWER_DLL` The dynamic link library is newer than the specified version.
1409	`MSK_RES_ERR_QCON_SUBI_TOO_LARGE` Invalid value in `qcsubi`.
1113	`MSK_RES_ERR_MPS_MUL_QSEC` Multiple `QSECTION`s are specified for a constraint in the MPS data file.
502	`MSK_RES_WRN_EMPTY_NAME` A variable or constraint name is empty. The output file may be invalid.
4003	`MSK_RES_TRM_MIO_NEAR_REL_GAP` The mixed-integer optimizer terminated because the near optimal relative gap tolerance was satisfied.
80	`MSK_RES_WRN_LP_OLD_QUAD_FORMAT` Missing '/2' after quadratic expressions in bound or objective.
1102	`MSK_RES_ERR_MPS_INV_MARKER` An invalid marker has been specified in the MPS file.
1070	`MSK_RES_ERR_NULL_NAME` An all blank name has been specified.
1264	`MSK_RES_ERR_NEGATIVE_APPEND` Cannot append a negative number.
1270	`MSK_RES_ERR_INV_SK` Invalid status key code.
1006	`MSK_RES_ERR_PROB_LICENSE` The software is not licensed to solve the problem.
1015	`MSK_RES_ERR_LICENSE_SERVER` The license server is not responding.
4005	`MSK_RES_TRM_USER_BREAK` The optimizer terminated due to a user break.

continued on next page

continued from previous page

400	`MSK_RES_WRN_TOO_FEW_BASIS_VARS` An incomplete basis has been specified. Too few basis variables are specified.
1161	`MSK_RES_ERR_WRITE_LP_NON_UNIQUE_NAME` An auto-generated name is not unique.
1108	`MSK_RES_ERR_MPS_INV_BOUND_KEY` An invalid bound key occurred in an MPS file.
1472	`MSK_RES_ERR_NAN_IN_BUX` u^x contains an invalid floating point value, i.e. a `NaN`.
1109	`MSK_RES_ERR_MPS_INV_SEC_NAME` An invalid section name occurred in an MPS file.
1266	`MSK_RES_ERR_BASIS` An invalid basis is specified. Either too many or too few basis variables are specified.
1257	`MSK_RES_ERR_INV_BKX` An invalid bound key is specified for a variable.
1002	`MSK_RES_ERR_LICENSE_VERSION` The license is valid for another version of MOSEK.
4006	`MSK_RES_TRM_STALL` The optimizer terminated due to slow progress. Normally there are three possible reasons for this: Either a bug in MOSEK, the problem is badly formulated, or, in case of nonlinear problems, the nonlinear call-back functions are incorrect. Please contact MOSEK support if this happens.
1048	`MSK_RES_ERR_THREAD_CREATE` Could not create a thread. This error may occur if a large number of environments are created and not deleted again. In any case it is a good practice to minimize the number of environments created.
1128	`MSK_RES_ERR_MPS_INVALID_OBJ_NAME` An invalid objective name is specified.
1217	`MSK_RES_ERR_PARAM_VALUE_STR` The parameter value string is incorrect.
1222	`MSK_RES_ERR_INDEX_ARR_IS_TOO_LARGE` An index in an array argument is too large.
1306	`MSK_RES_ERR_CONE_TYPE_STR` Invalid cone type specified.
1301	`MSK_RES_ERR_CONE_SIZE` A cone with too few members is specified.
1258	`MSK_RES_ERR_INV_VAR_TYPE`

continued on next page

continued from previous page

	An invalid variable type is specified for a variable.
1157	`MSK_RES_ERR_LP_FILE_FORMAT` Syntax error in an LP file.
1021	`MSK_RES_ERR_LICENSE_CANNOT_CONNECT` MOSEK cannot connect to the license server. Most likely the license server is not up and running.
4002	`MSK_RES_TRM_OBJECTIVE_RANGE` The optimizer terminated on the bound of the objective range.
1126	`MSK_RES_ERR_MPS_TAB_IN_FIELD3` A tab char occurred in field 3.
350	`MSK_RES_WRN_UNDEF_SOL_FILE_NAME` Undefined name occurred in a solution.
1255	`MSK_RES_ERR_INV_BK` Invalid bound key.
1014	`MSK_RES_ERR_FLEXLM` The FLEXlm license manager reported an error.
2550	`MSK_RES_ERR_MBT_INCOMPATIBLE` 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.
65	`MSK_RES_WRN_NAME_MAX_LEN` A name is longer than the buffer that is supposed to hold it.
1165	`MSK_RES_ERR_NAME_MAX_LEN` A name is longer than the buffer that is supposed to hold it.
1261	`MSK_RES_ERR_FIRST` Invalid `first`.
4031	`MSK_RES_TRM_INTERNAL_STOP` The optimizer terminated for internal reasons. Please contact MOSEK support.
1117	`MSK_RES_ERR_MPS_CONE_TYPE` Invalid cone type specified in a `CSECTION`.
1005	`MSK_RES_ERR_SIZE_LICENSE` The problem is bigger than the license.
1375	`MSK_RES_ERR_HUGE_C` A huge value in absolute size is specified for one c_j.
1446	`MSK_RES_ERR_UNDEFINED_OBJECTIVE_SENSE` The objective sense has not been specified before the optimization.
802	`MSK_RES_WRN_PRESOLVE_OUTOFSPACE` The presolve is incomplete due to lack of space.
1130	`MSK_RES_ERR_ORD_INVALID_BRANCH_DIR`

continued on next page

continued from previous page	
	An invalid branch direction key is specified.
2501	`MSK_RES_ERR_INV_MARKI` Invalid value in marki.
2502	`MSK_RES_ERR_INV_MARKJ` Invalid value in markj.
2500	`MSK_RES_ERR_NO_SOLUTION_IN_CALLBACK` The required solution is not available.
2900	`MSK_RES_ERR_INVALID_UTF8` An invalid UTF8 string is encountered.
1105	`MSK_RES_ERR_MPS_UNDEF_CON_NAME` An undefined constraint name occurred in an MPS file.
1012	`MSK_RES_ERR_SIZE_LICENSE_INTVAR` The problem contains too many integer variables to be solved with the available license.
1230	`MSK_RES_ERR_INF_DOU_NAME` A double information name is invalid.
1552	`MSK_RES_ERR_NO_OPTIMIZER_VAR_TYPE` No optimizer is available for this class of optimization problems.

D.37 Response code type

Value	Name Description
1	`MSK_RESPONSE_WRN` The response code is a warning.
2	`MSK_RESPONSE_TRM` The response code is an optimizer termination status.
4	`MSK_RESPONSE_UNK` The response code does not belong to any class.
0	`MSK_RESPONSE_OK` The response code is OK.
3	`MSK_RESPONSE_ERR` The response code is an error.

D.38 Scaling type

Value	Name Description
1	`MSK_SCALING_NONE` No scaling is performed.
2	`MSK_SCALING_MODERATE` A conservative scaling is performed.
3	`MSK_SCALING_AGGRESSIVE` A very aggressive scaling is performed.
0	`MSK_SCALING_FREE` The optimizer chooses the scaling heuristic.

D.39 Sensitivity types

Value	Name Description
1	`MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION` Optimal partition sensitivity analysis is performed.
0	`MSK_SENSITIVITY_TYPE_BASIS` Basis sensitivity analysis is performed.

D.40 Degeneracy strategies

Value	Name Description
0	`MSK_SIM_DEGEN_NONE` The simplex optimizer should use no degeneration strategy.
3	`MSK_SIM_DEGEN_MODERATE` The simplex optimizer should use a moderate degeneration strategy.
4	`MSK_SIM_DEGEN_MINIMUM` The simplex optimizer should use a minimum degeneration strategy.
2	`MSK_SIM_DEGEN_AGGRESSIVE` The simplex optimizer should use an aggressive degeneration strategy.
1	`MSK_SIM_DEGEN_FREE` The simplex optimizer chooses the degeneration strategy.

D.41 Hot-start type employed by the simplex optimizer

Value	Name Description
0	`MSK_SIM_HOTSTART_NONE` The simplex optimizer performs a coldstart.
2	`MSK_SIM_HOTSTART_STATUS_KEYS` Only the status keys of the constraints and variables are used to choose the type of hot-start.
1	`MSK_SIM_HOTSTART_FREE` The simplex optimize chooses the hot-start type.

D.42 Simplex selection strategy

Value	Name Description
1	`MSK_SIM_SELECTION_FULL` The optimizer uses full pricing.
5	`MSK_SIM_SELECTION_PARTIAL` The optimizer uses a partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.
0	`MSK_SIM_SELECTION_FREE` The optimizer chooses the pricing strategy.
2	`MSK_SIM_SELECTION_ASE` The optimizer uses approximate steepest-edge pricing.
3	`MSK_SIM_SELECTION_DEVEX` The optimizer uses devex steepest-edge pricing (or if it is not available an approximate steep-edge selection).
4	`MSK_SIM_SELECTION_SE` The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

D.43 Solution items

Value	Name Description
4	`MSK_SOL_ITEM_SUC` Lagrange multipliers for upper bounds on the constraints.
0	`MSK_SOL_ITEM_XC`

continued on next page

continued from previous page

	Solution for the constraints.
1	`MSK_SOL_ITEM_XX` Variable solution.
2	`MSK_SOL_ITEM_Y` Lagrange multipliers for equations.
5	`MSK_SOL_ITEM_SLX` Lagrange multipliers for lower bounds on the variables.
6	`MSK_SOL_ITEM_SUX` Lagrange multipliers for upper bounds on the variables.
7	`MSK_SOL_ITEM_SNX` Lagrange multipliers corresponding to the conic constraints on the variables.
3	`MSK_SOL_ITEM_SLC` Lagrange multipliers for lower bounds on the constraints.

D.44 Solution status keys

Value	Name Description
6	`MSK_SOL_STA_DUAL_INFEAS_CER` The solution is a certificate of dual infeasibility.
5	`MSK_SOL_STA_PRIM_INFEAS_CER` The solution is a certificate of primal infeasibility.
0	`MSK_SOL_STA_UNKNOWN` Status of the solution is unknown.
8	`MSK_SOL_STA_NEAR_OPTIMAL` The solution is nearly optimal.
12	`MSK_SOL_STA_NEAR_PRIM_INFEAS_CER` The solution is almost a certificate of primal infeasibility.
2	`MSK_SOL_STA_PRIM_FEAS` The solution is primal feasible.
15	`MSK_SOL_STA_NEAR_INTEGER_OPTIMAL` The primal solution is near integer optimal.
10	`MSK_SOL_STA_NEAR_DUAL_FEAS` The solution is nearly dual feasible.
14	`MSK_SOL_STA_INTEGER_OPTIMAL` The primal solution is integer optimal.
13	`MSK_SOL_STA_NEAR_DUAL_INFEAS_CER`

continued on next page

continued from previous page

Value	Name Description
	The solution is almost a certificate of dual infeasibility.
11	`MSK_SOL_STA_NEAR_PRIM_AND_DUAL_FEAS` The solution is nearly both primal and dual feasible.
1	`MSK_SOL_STA_OPTIMAL` The solution is optimal.
4	`MSK_SOL_STA_PRIM_AND_DUAL_FEAS` The solution is both primal and dual feasible.
9	`MSK_SOL_STA_NEAR_PRIM_FEAS` The solution is nearly primal feasible.
3	`MSK_SOL_STA_DUAL_FEAS` The solution is dual feasible.

D.45 Solution types

Value	Name Description
2	`MSK_SOL_ITG` The integer solution.
0	`MSK_SOL_ITR` The interior solution.
1	`MSK_SOL_BAS` The basic solution.

D.46 Solve primal or dual form

Value	Name Description
1	`MSK_SOLVE_PRIMAL` The optimizer should solve the primal problem.
2	`MSK_SOLVE_DUAL` The optimizer should solve the dual problem.
0	`MSK_SOLVE_FREE` The optimizer is free to solve either the primal or the dual problem.

D.47 String parameter types

Value	Name Description
8	`MSK_SPAR_PARAM_COMMENT_SIGN` Only the first character in this string is used. It is considered as a start of comment sign in the MOSEK parameter file. Spaces are ignored in the string.
3	`MSK_SPAR_FEASREPAIR_NAME_PREFIX` Not applicable.
0	`MSK_SPAR_BAS_SOL_FILE_NAME` Name of the `bas` solution file.
12	`MSK_SPAR_READ_MPS_OBJ_NAME` Name of the free constraint used as objective function. An empty name means that the first constraint is used as objective function.
5	`MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL` The constraint and variable associated with the total weighted sum of violations are each given the name of this parameter postfixed with `CON` and `VAR` respectively.
4	`MSK_SPAR_FEASREPAIR_NAME_SEPARATOR` Not applicable.
10	`MSK_SPAR_PARAM_WRITE_FILE_NAME` The parameter database is written to this file.
6	`MSK_SPAR_INT_SOL_FILE_NAME` Name of the `int` solution file.
14	`MSK_SPAR_READ_MPS_RHS_NAME` Name of the RHS used. An empty name means that the first RHS vector is used.
21	`MSK_SPAR_STAT_FILE_NAME` Statistics file name.
24	`MSK_SPAR_WRITE_LP_GEN_VAR_NAME` Sometimes when an LP file is written additional variables must be inserted. They will have the prefix denoted by this parameter.
1	`MSK_SPAR_DATA_FILE_NAME` Data are read and written to this file.
13	`MSK_SPAR_READ_MPS_RAN_NAME` Name of the RANGE vector used. An empty name means that the first RANGE vector is used.
17	`MSK_SPAR_SOL_FILTER_XC_LOW`

continued on next page

continued from previous page

	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.
18	`MSK_SPAR_SOL_FILTER_XC_UPR` A filter used to determine which constraints should be listed in the solution file. A value of “0.5” means that all constraints having `xc[i]<0.5` should be listed, whereas “-0.5” means all constraints having `xc[i]<=buc[i]-0.5` should be listed. An empty filter means that no filter is applied.
11	`MSK_SPAR_READ_MPS_BOU_NAME` Name of the BOUNDS vector used. An empty name means that the first BOUNDS vector is used.
20	`MSK_SPAR_SOL_FILTER_XX_UPR` A filter used to determine which variables should be listed in the solution file. A value of “0.5” means that all constraints having `xx[j]<0.5` should be printed, whereas “-0.5” means all constraints having `xx[j]<=bux[j]-0.5` should be listed. An empty filter means no filter is applied.
23	`MSK_SPAR_STAT_NAME` Name used when writing the statistics file.
9	`MSK_SPAR_PARAM_READ_FILE_NAME` Modifications to the parameter database is read from this file.
7	`MSK_SPAR_ITR_SOL_FILE_NAME` Name of the `itr` solution file.
15	`MSK_SPAR_SENSITIVITY_FILE_NAME` Not applicable.
2	`MSK_SPAR_DEBUG_FILE_NAME` MOSEK debug file.
22	`MSK_SPAR_STAT_KEY` Key used when writing the summary file.
16	`MSK_SPAR_SENSITIVITY_RES_FILE_NAME` Not applicable.
19	`MSK_SPAR_SOL_FILTER_XX_LOW`

continued on next page

continued from previous page

	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.

D.48 Status keys

Value	Name Description
2	`MSK_SK_SUPBAS` The constraint or variable is super basic.
1	`MSK_SK_BAS` The constraint or variable is in the basis.
5	`MSK_SK_FIX` The constraint or variable is fixed.
3	`MSK_SK_LOW` The constraint or variable is at its lower bound.
6	`MSK_SK_INF` The constraint or variable is infeasible in the bounds.
0	`MSK_SK_UNK` The status for the constraint or variable is unknown.
4	`MSK_SK_UPR` The constraint or variable is at its upper bound.

D.49 Starting point types

Value	Name Description
1	`MSK_STARTING_POINT_CONSTANT` The starting point is set to a constant. This is more reliable than a non-constant starting point.
0	`MSK_STARTING_POINT_FREE` The starting point is chosen automatically.

D.50 Stream types

Value	Name Description
1	`MSK_STREAM_MSG` Message stream.
3	`MSK_STREAM_WRN` Warning stream.
0	`MSK_STREAM_LOG` Log stream.
2	`MSK_STREAM_ERR` Error stream.

D.51 Integer values

Value	Name Description
1024	`MSK_MAX_STR_LEN` Maximum string length allowed in MOSEK.
20	`MSK_LICENSE_BUFFER_LENGTH` The length of a license key buffer.

D.52 Variable types

Value	Name Description
1	`MSK_VAR_TYPE_INT` Is an integer variable.
0	`MSK_VAR_TYPE_CONT` Is a continuous variable.

D.53 XML writer output mode

Value	Name Description
1	`MSK_WRITE_XML_MODE_COL` Write in column order.
0	`MSK_WRITE_XML_MODE_ROW`

continued on next page

continued from previous page

	Write in row order.

Bibliography

[1] Richard C. Grinold abd Ronald N. Kahn. *Active portfolio management.* McGraw-Hill, New York, 2 edition, 2000.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. Network flows. In G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, editors, *Optimization*, volume 1, pages 211–369. North Holland, Amsterdam, 1989.

[3] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Math. Programming*, 95(1):3–51, 2003.

[4] E. D. Andersen and K. D. Andersen. Presolving in linear programming. *Math. Programming*, 71(2):221–245, 1995.

[5] E. D. Andersen and K. D. Andersen. The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In J. B. G. Frenk, C. Roos, T. Terlaky, and S. Zhang, editors, *High Performance Optimization Techniques, Proceedings of the HPOPT-II conference*, 1997. forthcoming.

[6] E. D. Andersen, J. Gondzio, Cs. Mészáros, and X. Xu. Implementation of interior point methods for large scale linear programming. In T. Terlaky, editor, *Interior-point methods of mathematical programming*, pages 189–252. Kluwer Academic Publishers, 1996.

[7] E. D. Andersen, C. Roos, and T. Terlaky. On implementing a primal-dual interior-point method for conic quadratic optimization. *Math. Programming*, 95(2), February 2003.

[8] E. D. Andersen and Y. Ye. Combining interior-point and pivoting algorithms. *Management Sci.*, 42(12):1719–1731, December 1996.

[9] E. D. Andersen and Y. Ye. A computational study of the homogeneous algorithm for large-scale convex optimization. *Computational Optimization and Applications*, 10:243–269, 1998.

[10] E. D. Andersen and Y. Ye. On a homogeneous algorithm for the monotone complementarity problem. *Math. Programming*, 84(2):375–399, February 1999.

[11] K. D. Andersen. A Modified Schur Complement Method for Handling Dense Columns in Interior-Point Methods for Linear Programming. *ACM Trans. Math. Software*, 22(3):348–356, 1996.

[12] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming: Theory and algorithms.* John Wiley and Sons, New York, 2 edition, 1993.

[13] C. Beightler and D. T. Phillips. *Applied geometric programming.* John Wiley and Sons, New York, 1976.

[14] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Math. Programming*, 88(3):411–424, 2000.

[15] A. Ben-Tal and A Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications.* MPS/SIAM Series on Optimization. SIAM, 2001.

[16] S.P. Boyd, S.J. Kim, L. Vandenberghe, and A. Hassibi. A Tutorial on Geometric Programming. Technical report, ISL, Electrical Engineering Department, Stanford University, Stanford, CA, 2004. Available at `http://www.stanford.edu/~boyd/gp_tutorial.html`.

[17] V. Chvátal. *Linear programming.* W.H. Freeman and Company, 1983.

[18] N. Gould and P. L. Toint. Preprocessing for quadratic programming. *Math. Programming*, 100(1):95–132, 2004.

[19] J. L. Kennington and K. R. Lewis. Generalized networks: The theory of preprocessing and an emperical analysis. *INFORMS Journal on Computing*, 16(2):162–173, 2004.

[20] M. S. Lobo, L. Vanderberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra Appl.*, 284:193–228, November 1998.

[21] M. S. Lobo, M. Fazel, and S. Boyd. Portfolio optimization with linear and fixed transaction costs. Technical report, CDS, California Institute of Technology, 2005. To appear in Annals of Operations Research. `http://www.cds.caltech.edu/~maryam/portfolio.html`.

[22] J. L. Nazareth. *Computer Solution of Linear Programs.* Oxford University Press, New York, 1987.

[23] C. Roos, T. Terlaky, and J. -Ph. Vial. *Theory and algorithms for linear optimization: an interior point approach.* John Wiley and Sons, New York, 1997.

[24] Bernd Scherer. *Portfolio construction and risk budgeting.* Risk Books, 2 edition, 2004.

[25] R. J. Vanderbei. *Linear Programming. Foundations and Extensions.* Kluwer Academic Publishers, Boston/London/Dordrect, 1997.

[26] S. W. Wallace. Decision making under uncertainty: Is sensitivity of any use. *Oper. Res.*, 48(1):20–25, January 2000.

[27] H. P. Williams. *Model building in mathematical programming.* John Wiley and Sons, 3 edition, 1993.

[28] L. A. Wolsey. *Integer programming.* John Wiley and Sons, 1998.

Index