MOSEK Rmosek Package

Release 8.0.0.60

MOSEK ApS

2017

CONTENTS

CHAPTER

ONE

INTRODUCTION

The **MOSEK** Optimization Suite 8.0.0.60 is a powerfull software package capable of solving large-scale optimization problems of the following kind:

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

Integer constrained variables are supported for all problem classes except for semidefinite and general convex problems. In order to obtain an overview of features in the **MOSEK** Optimization Suite consult the product introduction guide.

1.1 Why the Rmosek Package?

The Rmosek Package provides access to most functionalities of **MOSEK** from an R-language software environment. The package is adjusted to the typical R user.

1.2 License agreement

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

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

zlib

MOSEK includes the *zlib* library obtained from the zlib website. The license agreement for *zlib* is shown in Listing 1.1.

Listing 1.1: *zlib* license.

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

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

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

fplib

MOSEK includes the floating point formatting library developed by David M. Gay obtained from the netlib website. The license agreement for *fplib* is shown in Listing 1.2.

Listing 1.2: *fplib* license.

CHAPTER

TWO

INSTALLATION

2.1 Windows Platforms

The interface documented here is part of the Rmosek package distributed from http://rmosek.r-forge.r-project.org/. Notice, however, that a pre-compiled binary version of it has not been distributed. Thus a small amount of configuration is necessary in order to install the package. It is not very difficult, but besides the basic installation of R you will need two pieces of software readily available through the Internet. In summary:

- **MOSEK** (the optimization library we interface to.
- Rtools (the tools needed for R package development

The following is a step-by-step guide through the installation of the package. We skip the installation of **MOSEK** as it has been well documented elsewhere and assume that the target machine already has a working installation. Please refer to the documentation available at https://mosek.com/resources/doc/ for the verification of this installation.

For additional help installing this interface, the section on packages in the R for Windows *FAQ* available on the CRAN website may be useful. The *R Installation and Administration* manual, also published by CRAN on their website, is another good source of information.

2.1.1 Setting Up the Target Machine

Assuming a working installation of R and **MOSEK** on the machine targeted for the R-to-**MOSEK** interface, the first step is to download Rtools for Windows. From this program you will need to install the component called *R toolset*, the *Cygwin DLLs*, and the R toolchain.

Note: R toolset and Cygwin DLLs will extend the Windows CMD with Unix-style commands, while the R toolchain (based on MinGW compilers) makes it possible to compile the C++ source code in the package. These components can be replaced by any other Unix-style shell and C++ build chain, but in the remaining guide the use of Rtools will be assumed.

After the installation of Rtools you will have to set the Windowsenvironment variable called `PATH`, in order to utilize the components. Assuming that the home directory of the Rtools installation was `C:\Rtools`, the entries shown below will have to be added to the existing `PATH` variable. Note that all entries in the `PATH` variable must be separated by a semicolon (;), and that all these entries have to represent folders that exist on the target machine.

- Add `C:\Rtools\bin;` to enable the R toolset and Cygwin DLLs.
- Add `C:\Rtools\gcc-`*`VERSION`*`bin;` (for some *VERSION*) to enable the R toolchain.

That was it, but before we move on to the installation process, please ensure that the PATH variable also contains the `bin` folder of all **MOSEK** installations (32 and/or 64 bit) that you wish this interface to target. This is necessary for automatic configuration to work, and could look something like:

```
C:\Program\Mosek\8\tools\platform\win64x86\bin
```

2.1.2 Installing the Package with Automatic Configuration

Automatic configuration works equivalently to calling `where mosek` in the Windows CMD. It searches the environment variable called `PATH` for a folder with an executable called `mosek`. Note that if more than one such folder exists, only the one mentioned first in the `PATH` variable is chosen. It then determines the most ordinary of the available optimization libraries within this folder (typically `mosek.lib` or `mosek64.lib`), along with other relevant information. This configuration should work for all users installing the package on a single architecture (64 or 32 bit) and only requiring the ordinary optimization library. Otherwise, manual configuration of the package will be needed.

Now open R for the architecture (64 or 32 bit) you wish to install the package on. Make sure all your packages are up to date by writing `update.packages()`, and execute a command similar to the one shown below. This will install the Rmosek package:

```
install.packages("Rmosek", type="source", INSTALL_opts="--no-multiarch", repos="http://
↪download.mosek.com/R/8")
```

Note that this package will have to be installed from *source* as it needs a static link to the **MOSEK** optimization library. This unfortunately means that dependencies (i.e. the *Matrix* package) will also be installed from source if new releases are available. Since it is more time-consuming to install the *Matrix* package from source, it is recommended to start with a call to `update.packages()`. The availability of the *Matrix* package should not be a problem, as it has been part of the standard R installation since version 2.9.0.

2.1.3 Installing the Package with Manual Configuration

If the automatic configuration does not suit your particular needs, or fails for some reason, a manual configuration may work instead. Unfortunately the `configure.vars` parameter of the `install.packages` command does not work on Windows, meaning that the files of the Rmosek source package will have to be edited. This can be difficult for non-savvy users, but have hopefully been documented sufficiently here.

How to manually configure the Rmosek package?

When you download the package from http://rmosek.r-forge.r-project.org/, it comes in a compressed archive called `RMOSEK_`*`VERSION`*`.tar.gz` for some version number `VERSION`. In order to configure the package, you will have to go through the following steps. A more thorough explanation of each step will be given afterwards.

1. Extract the archive into a directory.
2. Setup the local system descriptors, `Localsys.txt`, for each of the sub-architectures you will be using: 64 bit (e.g. x64) and/or 32 bit (e.g. i386). These files come with a guide written into them that clearly states how this should be done.
3. Compress the directory back into an archive.

The first step is to extract the `rmosek_`*`VERSION`*`.tar.gz` archive that you downloaded. This can either be done by using one of the many tools freely available online, or the `tar` command that was installed with Rtools. If you choose to use the `tar` command, you can extract the package by opening Windows CMD and executing a command similar to:

```
tar --no-same-owner -zxvf LOCATION/RMOSEK_VERSION.tar.gz
```

Remember to exchange *LOCATION* and *VERSION* with the correct values. For those not familiar with Windows CMD, we recommend the use of an external tool as this configuration can then be performed entirely within Windows Explorer.

The second step is to tell the package where to find the **MOSEK** optimization library. Open the extracted directory called `Rmosek` in either Windows CMD or Windows Explorer, and navigate to the subdirectory called `src\setup`. If you want to install a 64 bit version of the package (making an interface between 64 bit R and 64 bit **MOSEK**), open the folder `x64` and follow the guide in `Localsys.txt`. If you want to install a 32 bit version of the package (making an interface between 32 bit R and 32 bit **MOSEK**), open instead the folder `i386` and follow the guide in this `Localsys.txt`. An example of this is shown below

Listing 2.1: An exampel of local system descriptor.

```
##
## Greetings user of the R-to -MOSEK interface !
##
## If you are sitting on a WINDOWS 64 bit platform , this is the file that
## you will have to setup before this package can be installed .
## (see e.g. the R-to -MOSEK userguide )
##
##################
## Step 1 of 2 ##
##################
## Please substitute [ MOSEK_HOME_PATH ] below , with the path to the
## platform - specific folder within the MOSEK installation you want to
## use. Note that this path should contain a "bin" and a "h" folder .
## -----------------------------------------------------------------
## For example you can write:
## PKG_MOSEKHOME =C:\ Progra ~1\ Mosek \8\ tools\ platform \ win64x86
##
## If your computer contains the two directories :
## C:\ Progra ~1\ Mosek \8\ tools\ platform \ win64x86 \bin
## C:\ Progra ~1\ Mosek \8\ tools\ platform \ win64x86 \h
## -----------------------------------------------------------------
PKG_MOSEKHOME =[ MOSEK_HOME_PATH ]
##################
## Step 2 of 2 ##
##################
## Please substitute [ MOSEK_LIB_FILE ] below , with the name of the library
## you wish to use within the "bin" folder of your PKG_MOSEKHOME path.
## This "bin" folder must contain a file called [ MOSEK_LIB_FILE ]. lib.
## -----------------------------------------------------------------
## Continuing the example from above , you can write:
## PKG_MOSEKLIB = mosek64_8_0
##
## If your computer contains the file:
## C:\ Progra ~1\ Mosek \8\ tools\ platform \ win64x86 \bin\ mosek64_8_0 .lib
## -----------------------------------------------------------------
PKG_MOSEKLIB =[ MOSEK_LIB_FILE ]
```

The third and final step is to compress the (previously extracted and now altered) directory called `Rmosek`, back into the `Rmosek_VERSION.tar.gz` archive. Again you can either make use of the external tools from step one, or open the Windows CMD and execute a command similar to:

```
tar -zcvf Rmosek_VERSION.tar.gz Rmosek`
```

How to install the manually configured Rmosek package?

Open R, either the console or the graphical user interface, for the architecture (64 or 32 bit) you wish to install the package on. Make sure all your packages are up to date by writing `update.packages()`, and execute the following command to install the Rmosek package::

```
install.packages("LOCATION/Rmosek_VERSION.tar.gz", repos=NULL, type="source", INSTALL_opts="--
↪no-multiarch")
```

Remember to exchange *LOCATION* and *VERSION* with the correct values. The first argument should be the path to your manually configured package. The second argument tells that the package is local and not in an online repository. The third argument tells that it is a source package and so should be compiled. The fourth and final argument specifies that you only wish to install the package to the specific architecture (64 or 32 bit) of the opened R program. If you wish to install on both architectures, and did configure the package for both 32 and 64 bit, simply remove this last argument.

Notice that if you wish to uninstall the Rmosek package at some point, this can be done as for any other package with the command `remove.packages(Rmosek)`.

2.2 UNIX-alike Platforms

The interface documented here is part of the Rmosek package distributed from http://rmosek.r-forge.r-project.org/. Notice, however, that a pre-compiled binary version of it has not been distributed (Such a binary would have to be built individually for each version of R and **MOSEK**). Thus a small amount of configuration is necessary in order to install the package. It is not very difficult, but besides the basic installation of R you will need **MOSEK** (the optimization library we interface to) readily available through the Internet.

The following is a step-by-step guide through the installation of the package. We skip the installation of **MOSEK** as it has been well documented elsewhere and assume that the target machine already has a working installation. Please refer to the documentation available at https://mosek.com/resources/doc/ for the verification of this installation.

For additional help on installing this interface, the section on installing packages in the manual *R Installation and Administration*, published on the CRAN website, is a good source of information.

2.2.1 Setting Up the Target Machine

We assume here that you have a working installation of R and **MOSEK** on the machine targeted for the R-to-**MOSEK** interface. The architectures (32 or 64 bit) of these two programs must be exactly the same for consistency. We further assume that the target machine have a build chain (e.g. `GCC` compilers for C/C++) installed.

For automatic configuration to work, the `PATH` variable should contain the `bin` folder of the single **MOSEK** installation (32 or 64 bit) that you wish this interface to target. This could look something like:

```
~/mosek/8/tools/platform/linux64x86/bin
```

If more than one `bin` folder from a **MOSEK** installation are specified, only the first one will be found by automatic configuration. Additional architectures can be added afterwards if necessary.

2.2.2 Installing the Package with Automatic Configuration

Automatic configuration works equivalently to calling `which mosek` ina terminal window. It searches the environment variable called `PATH` for a folder with an executable called `mosek`. Note that if more than one such folder exists, only the one mentioned first in the `PATH` variable is chosen. It then determines the most ordinary of the available optimization libraries within this folder (typically `libmosek` or `libmosek64` with the extension `.so` or `.dylib`), along with other relevant information. This configuration should work for all users only installing the package on a single architecture (64 or 32 bit) and only requiring the ordinary optimization library. Otherwise, manual configuration of the package will be needed.

Now open R for the architecture (64 or 32 bit) you wish to install the package on. Make sure all your packages are up to date by writing `update.packages()`, and execute a command similar to the one shown below. This will install the Rmosek package::

```
install.packages("Rmosek", type="source", INSTALL_opts="--no-multiarch", repos="http://
↪download.mosek.com/R/8")
```

Note that this package will have to be installed from *source* as it needs a static link to the **MOSEK** optimization library. This unfortunately means that dependencies (i.e. the 'Matrix' package) will also be installed from source if new releases are available. Since it is more time-consuming to install the *Matrix* package from source, it is recommended to start with a call to `update.packages()`. The availability of the *Matrix* package should not be a problem, as it has been part of the standard R installation since version 2.9.0.

2.2.3 Installing the Package with Manual Configuration

Open R for one of the architectures (64 or 32 bit) you wish to install the package on. Make sure all your packages are up to date by writing `update.packages()`, and execute a command similar to the one shown below, with a correct definition of `PKG_MOSEKHOME` and `PKG_MOSEKLIB`. This will install the Rmosek package:

Remember to exchange the `...` of both `PKG_MOSEKHOME` and `PKG_MOSEKLIB` with the correct values, explained as follows. The definition of the first argument, `PKG_MOSEKHOME`, should be the folder in your **MOSEK** installation, containing a `bin` and `h` subdirectory for the platform and architecture matching that of the opened R program. This could for instance look something like:

```
/home/username/mosek/8/tools/platform/linux64x86
```

Notice that auto-expansions such as ∼ does not work, and in case the folder definition contains spaces you will either have to wrap the definition in single-quotes or add backslashes in front of all spaces.

The definition of the argument `PKG_MOSEKLIB` should be the name of the optimization library in the `bin` subdirectory that you wish to utilize in the Rmosek package. This library will be statically linked to the package after a successful installation. Note that the name of the optimization library should be specified without the *lib* prefix, and without its file-extension. The `PKG_RMOSEKLIB` would thus normally be either `mosek` or `mosek64` (linking to respectively `libmosek.so` and `libmosek64.so`, or respectively `libmosek.dylib` and `libmosek64.dylib`, depending on the Unix-alike system). Using `mosek64` requires a 64 bit version of the opened R program and **MOSEK** installation, while `mosek` implies 32 bit.

How to install on multiple architectures?

If you wish to install the Rmosek package on multiple sub-architectures, you will first have to follow the above guide and install the package on one of the architectures (e.g. 32 or 64 bit). Afterwards as explained here, it can then be extended to other sub-architectures. This is also explained in *R Installation and Administration* (published on the CRAN website), in the subsection *Multiple sub-architectures* under *Installing packages*. For this to work you will need a preinstalled version of R and **MOSEK** in all the sub-architectures you wish this package to work with.

Open R in the sub-architecture you wish to extend your installation to, and execute a command similar to the one shown below::

```
install.packages("Rmosek", type="source", libs_only=TRUE, repos="http://download.mosek.com/R/8
↪", configure.vars="PKG_MOSEKHOME=... PKG_MOSEKLIB=...")
```

Remember to exchange ... of the variables `PKG_MOSEKHOME` and `PKG_RMOSELIB`, so that declared **MOSEK** installation always correspond to the current sub-architecture of the opened R program.

How to install from an offline location?

This is almost the same a described above except that you would have to add the argument `repos=NULL` to tell R that it should not use an online repository. Also, instead of writing the package name Rmosek, you should write the entire file-location of the package source file which should look something like `LOCATION/Rmosek_`*`VERSION`*`.tar.gz`.

What are the command-line equivalents?

Sometimes you do not want to open R for all sub-architectures, but instead perform the installation directly from the console. Notice though, that this will require you to download the package source as you will only be able to install from an offline location in this way. In the two commands below you will have to replace `SUBARCH_PATH`, `DIR` and `VERSION` with the correct values, but doing so should be straight forward. Notice that on newer versions of R (>= 2.12), you can also use the call `R --arch SUBARCH`, instead of specifying the `SUBARCH_PATH`. Remember to exchange `...` of the variables `PKG_MOSEKHOME` and `PKG_MOSEKLIB`, with the correct values as previously explained.

For the first architecture::

```
SUBARCH_PATH/R CMD INSTALL DIR/Rmosek_VERSION.tar.gz --no-multiarch --configure-vars="PKG_
↪MOSEKHOME=... PKG_MOSEKLIB=..."
```

For subsequent architectures::

```
SUBARCH_PATH/R CMD INSTALL DIR/Rmosek_VERSION.tar.gz --libs-only --configure-vars="PKG_
↪MOSEKHOME=... PKG_MOSEKLIB=..."
```

CHAPTER

THREE

GUIDELINES

3.1 Parallel optimization Using the The Multicore Package

The R package called *multicore*, provides functions for parallel execution of R code on machines with multiple cores or CPUs. Windows is not currently supported, but the package should work on most UNIX-alike platforms.

The multicore package works by copying the full memory state of the R session to new processes. While this seems like a large overhead, in practice, the copy is delayed until modification assuring a smooth parallel execution. The downside is that this low-level memory state copy is not safe for all types of resources. As an example, parallel interactions with the GUI or on-screen devices can cause the R session to crash. It is thus recommended only to use the multicore package in console R.

In the Rmosek package a license is an externally acquired resource, and attempts to simply copy the memory state of this resource will provoke a session crash. Thus, licenses should always be released before the time of parallelization.

Note: Always call `mosek_clean()` before a parallelizing operator. Failure to do so is likely to provoke session crashes.

A consequence of this is that each new process will be using a separate license. That is, your license system should allow 8 licenses to be checked out simultaneously, if you wish to solve 8 optimization problems in parallel. Please note that unlimited academic and commercial licenses are available at **MOSEK**.

3.2 The license system

MOSEK is a commercial product that **always** needs a valid license to work. A license is typically provided as a license file that allows the user to access the subset of the **MOSEK** Optimization Suite functionalities it is entitled for, and for the right amount of time. **MOSEK** uses a third party license manager to implement license checking.

By default a license token remains checked out for the duration of the **MOSEK** session, i.e.

1. a license token is checked out when *mosek* is called the first time and
2. it is returned when R is terminated, or calling *mosek_clean*.

To change the license systems behavior to returning the license token after each call to **MOSEK** set the parameter *MSK_IPAR_CACHE_LICENSE* to *MSK_OFF*.

```
prob$iparam <- list(CACHE_LICENSE = "MSK_OFF");
```

3.2.1 Waiting for a free license

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

```
prob$iparam <- list(LICENSE_WAIT = "MSK_ON");
```

3.2.2 Manually stopping the license system

The license can be manually released by calling *mosek_clean*.

CHAPTER

FOUR

BASIC TUTORIALS

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

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

4.1 The Basics Tutorial

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

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

Listing 4.1: A simple script that reads a problem from file and solves it.

```
simple <- function( filename)
{

  r <- mosek_read(filename, list(usesol=FALSE, useparam=TRUE))

  if (identical(r$response$code, 0)) {

    print("Successfully read the optimization model")
    prob <- r$prob

    r <- try(mosek(prob, list(verbose=0)), silent=TRUE)
    if (inherits(r, "try-error")) {
      stop("Rmosek failed somehow!")
    }

    if (!identical(r$response$code, 0)) {
      cat(paste("**", "Response code:", r$response$code, "\n"))
      cat(paste("**", r$response$msg, "\n"))
      cat("Trying to continue..\n")
    }

    isdef <- try({
      rbas <- r$sol$bas;
      rbas$solsta;  rbas$prosta;  rbas$xx;
    }, silent=TRUE)
    if (inherits(isdef, "try-error")) {
      stop("Basic solution was incomplete!")
    }

    switch(rbas$solsta,
      OPTIMAL = {
        cat("The solution was optimal, I am happy!\n")
      },
      NEAR_OPTIMAL = {
        cat("The solution was close to optimal, very good..\n")
      },
      #OTHERWISE:
      {
        cat(paste("**", "Solution status:", rbas$solsta, "\n"))
        cat(paste("**", "Problem status:", rbas$prosta, "\n"))
        stop("Solution could not be accepted!")
      }
    )
  }
}
```

4.2 Linear Optimization

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

Minimize or maximize the objective function

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

subject to the linear constraints

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

and the bounds

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

where we have used the problem elements:

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

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

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

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

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

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

4.2.1 Example LO1

The following is an example of a linear optimization problem:

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

having the bounds

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

This is easily programmed in R as shown in Listing 4.2. The first line overwrites any previous definitions of the variable *lo1*, preparing for the new problem description. The problem is then defined and finally solved on the last line.

Listing 4.2: R implementation of problem (4.1).

```
get_lo1_solution_variables <- function(maxtime) {
  lo1 <- list(sense="max", c=c(3,1,5,1))
  lo1$A <- Matrix(c(3,1,2,0,2,1,3,1,0,2,0,3),
              nrow=3, byrow=TRUE, sparse=TRUE)
  lo1$bc <- rbind(blc=c(30,15,-Inf), buc=c(30,Inf,25));
  lo1$bx <- rbind(blx=c(0,0,0,0),    bux=c(Inf,10,Inf,Inf));
  lo1$dparam <- list(OPTIMIZER_MAX_TIME=maxtime)

  r <- try(mosek(lo1, list(verbose=0)), silent=TRUE)
  if (inherits(r, "try-error")) {
    stop("Rmosek failed somehow!")
  }

  if (!identical(r$response$code, 0)) {
    cat(paste("**", "Response code:", r$response$code, "\n"))
    cat(paste("**", r$response$msg, "\n"))
    cat("Trying to continue..\n")
  }

  isdef <- try({
    rbas <- r$sol$bas;
    rbas$solsta;  rbas$prosta;  rbas$xx;
  }, silent=TRUE)
  if (inherits(isdef, "try-error")) {
    stop("Basic solution was incomplete!")
  }

  switch(rbas$solsta,
    OPTIMAL = {
      cat("The solution was optimal, I am happy!\n")
    },
    NEAR_OPTIMAL = {
      cat("The solution was close to optimal, very good..\n")
    },
    #OTHERWISE:
    {
      cat(paste("**", "Solution status:", rbas$solsta, "\n"))
      cat(paste("**", "Problem status:", rbas$prosta, "\n"))
      stop("Solution could not be accepted!")
    }
  )
  return(rbas$xx)
}
```

Notice how the R value *Inf* is used in both the constraint bounds (*blc* and *buc*) and the variable upper bound (*bux*), to avoid the specification of an actual bound.

From this example the input arguments for the linear program follows easily.

- **Objective** The string is the objective goal and could be either *minimize*, *min*, *maximize* or *max*. The dense numeric vector specifies the coefficients in front of the variables in the linear objective function, and the optional constant scalar (reads: c zero) is a constant in the objective corresponding to c^f in problem , that will be assumed zero if not specified.
- **Constraint Matrix** The sparse matrix is the constraint matrix of the problem with the constraint coefficients written row-wise. Notice that for larger problems it may be more convenient to define an empty sparse matrix and specify the non-zero elements one at a time $A(i, j) = a_{ij}$, rather than writing out the full matrix as done in the *lo1* example. E.g. `Matrix(0,nrow=30,ncol=50,sparse=TRUE)`.
- **Bounds** The constraint bounds with rows *blc* (constraint lower bound) and *buc* (constraint upper

bound), as well as the variable bounds with rows *blx* (variable lower bound) and *bux* (variable upper bound), are both given as dense numeric matrices. These are equivalent to the bounds of problem , namely l^c, u^c, l^x and u^x.

4.3 Conic Quadratic Optimization

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

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

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

MOSEK can solve conic quadratic optimization problems of the form

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

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

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

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

- Quadratic cone:

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

- Rotated quadratic cone:

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

From these definition it follows that

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

is equivalent to

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

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

4.3.1 Example CQO1

We want to solve the following Conic Optimization Problem problem:

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

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

The first cone is of the quadratic cone type (*MSK_CT_QUAD*), while the second is of the rotated quadratic cone type (*MSK_CT_RQUAD*.) The subindexes of the variables used to define these cones follow naturally as seen in the Listing 4.3.

Listing 4.3: R implementation of model (4.2)

```
cqo1 <- list(sense = "min")
cqo1$c <- c(0,0,0,1,1,1)
cqo1$A <- Matrix(c(1,1,2,0,0,0),
                 nrow=1, byrow=TRUE, sparse=TRUE)
cqo1$bc <- rbind(blc = 1, buc = 1)
cqo1$bx <- rbind(blx = c(0,0,0,-Inf,-Inf,-Inf),
                 bux = rep(Inf,6))
cqo1$cones <- cbind(
  list("QUAD", c(4,1,2)),
  list("RQUAD", c(5,6,3))
)
rownames(cqo1$cones) <- c("type","sub");
r <- mosek(cqo1)
```

From this example the input arguments for a conic program follow easily. The objective function, the linear constraints and variable bounds should all be specified as for linear programs, and the only addition to this is the quadratic cones specified in the list-typed matrix.

The *cones* matrix has a column for each cone, and a row for each descriptive element. The first row called , should specify the cone type in a string, being either quadratic `QUAD` or rotated quadratic `RQUAD`. Notice that the **MOSEK** library cone type prefix `MSK_CT_` is optional. The second row called , should specify the subset of variables belonging to the cone in a numeric vector - and the ordering does matter! The i'th element of *sub* will be the index of the variable referred by x_i, in the cone definitions and . As an example, the rotated quadratic cone with subindexes `c(4,6,2,3)` would define the cone

$$\mathcal{K}_t = \left\{x \in \mathbb{R}^4 \ : \ 2x_4x_6 \geq x_2^2 + x_3^2, \ x_4 \geq 0, \ x_6 \geq 0\right\}.$$

Listing 4.4 showed a simple way to specify cones given an explicit representation. In many practical cases, however, cones are more conveniently specified in chunks or within a loop. For this purpose, preallocation should always be preferred as shown here.

Listing 4.4: R implementation of model (4.2) preallocating cones.

```
NUMCONES <- 2
cqo1$cones <- matrix(list(), nrow=2, ncol=NUMCONES)
  rownames(cqo1$cones) <- c("type","sub")
cqo1$cones[,1] <- list("QUAD", c(5,1,3))
cqo1$cones[,2] <- list("QUAD", c(6,2,4))
```

4.4 Semidefinite Optimization

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

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

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

MOSEK can solve semidefinite optimization problems of the form

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

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

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

4.4.1 Example SDO1

The problem

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

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

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

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

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

subject to the two linear constraints

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

and

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

and can be modeled in R as shown in Listing 4.5.

Listing 4.5: R implementation of model (4.3).

```
sdo1 <- list(sense="min")
sdo1$c     <- c(1,0,0)
sdo1$A     <- Matrix(c(1,0,0,
                       0,1,1), nrow=2, byrow=TRUE, sparse=TRUE)
sdo1$bc    <- rBind(blc = c(1, 0.5), buc = c(1, 0.5))
sdo1$bx    <- rBind(blx = rep(-Inf,3), bux = rep(Inf,3))
sdo1$cones <- cBind(list("quad", c(1,2,3)))
```

```
# One semidefinite matrix variable size 3x3:
N <- 3
sdo1$bardim <- c(N)

# Block triplet format specifying the lower triangular part
# of the symmetric coefficient matrix 'barc':
sdo1$barc$j <- c(1, 1, 1, 1, 1)
sdo1$barc$k <- c(1, 2, 3, 2, 3)
sdo1$barc$l <- c(1, 2, 3, 1, 2)
sdo1$barc$v <- c(2, 2, 2, 1, 1)

# Block triplet format specifying the lower triangular part
# of the symmetric coefficient matrix 'barA':
sdo1$barA$i <- c(1, 1, 1, 2, 2, 2, 2, 2, 2)
sdo1$barA$j <- c(1, 1, 1, 1, 1, 1, 1, 1, 1)
sdo1$barA$k <- c(1, 2, 3, 1, 2, 3, 2, 3, 3)
sdo1$barA$l <- c(1, 2, 3, 1, 2, 3, 1, 1, 2)
sdo1$barA$v <- c(1, 1, 1, 1, 1, 1, 1, 1, 1)

r <- mosek(sdo1)
barx   <- 1.0 * bandSparse(N, k=0:(1-N), symm=TRUE)
barx@x <- r$sol$itr$barx[[1]]
```

4.5 Quadratic Optimization

MOSEK can solve quadratic and quadratically constrained convex problems. This class of problems can be formulated as follows:

$$\begin{array}{lccccl} \text{minimize} & & & \frac{1}{2}x^TQ^ox + c^Tx + c^f & & \\ \text{subject to} & l_k^c & \leq & \frac{1}{2}x^TQ^kx + \sum_{j=0}^{n-1} a_{k,j}x_j & \leq & u_k^c, \quad k = 0,\ldots,m-1, \\ & l_j^x & \leq & x_j & \leq & u_j^x, \quad j = 0,\ldots,n-1. \end{array} \tag{4.4}$$

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

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

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

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

$$l_k^c \leq \frac{1}{2}x^TQ^kx + \sum_{j=0}^{n-1} a_{k,j}x_j \tag{4.5}$$

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

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

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

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

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

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

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

4.5.1 Example: Quadratic Objective

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

For the example (4.6) implies that

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

and that

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

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

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

Please note that there are quadratic terms in both constraints. This problem can be solved using *mosek* as reported in Listing 4.6.

Listing 4.6: Script implementing problem (4.6)

```
qo1 <- list()
qo1$sense <- "min"
qo1$c <- c(0,-1,0)
qo1$A <- Matrix(c(1,1,1), nrow=1, byrow=TRUE, sparse=TRUE)
qo1$bc <- rbind(blc = 1,
                buc = Inf)
qo1$bx <- rbind(blx = rep(0,3),
                bux = rep(Inf,3))

qo1$qobj <- list(i = c(1,  3,   2, 3),
                 j = c(1,  1,   2, 3),
                 v = c(2, -1, 0.2, 2))

r <- mosek(qo1)
```

4.6 Integer Optimization

An optimization problem where one or more of the variables are constrained to integer values is denoted an integer optimization problem.

Section *4.6.2* shows *how to input an initial feasible solution* to help the solver.

4.6.1 Example MILO1

In this section the example

$$\begin{array}{lll} \text{maximize} & x_0 + 0.64x_1 & \\ \text{subject to} & 50x_0 + 31x_1 & \leq 250, \\ & 3x_0 - 2x_1 & \geq -4, \\ & x_0, x_1 \geq 0 & \text{and integer} \end{array} \tag{4.7}$$

is used to demonstrate how to solve a problem with integer variables.

The example (4.7) is almost identical to a linear optimization problem (see *4.2*) except for some variables being integer constrained. Therefore, only the specification of the integer constraints requires something new compared to the linear optimization problem discussed previously.

This is easily programmed in R using the piece code shown in Listing 4.7,

Listing 4.7: R implementation of problem (4.7).

```
milo1 <- list(sense = "max")
milo1$c <- c(1, 0.64)
milo1$A <- Matrix(c( 50, 31 ,
                     3, -2 ),
                  nrow  = 2,
                  byrow = TRUE,
                  sparse= TRUE)
milo1$bc <-
  rbind(blc = c(-Inf,-4),
        buc = c(250,Inf))
milo1$bx <-
  rbind(blx = c(0,0),
        bux = c(Inf,Inf))
milo1$intsub <- c(1,2)
r <- mosek(milo1)
```

where x_1 and x_2 are pointed out as integer variables.

The input arguments follow those of a linear or conic program with the additional identification of the integer variables. The column vector `intsub` should simply contain indexes to the subset of variables for which integrality is required. For instance if x should be a binary $\{0, 1\}$-variable, its index in the problem formulation should be added to `intsub`, and its bounds $0 \leq x \leq 1$ should be specified explicitly.

If executed correctly you should be able to see the log of the interface and optimization process printed to the screen. The output structure will only include an integer solution `int`, since we are no longer in the continuous domain for which the interior-point algorithm operates. The structure also contains the problem status as well as the solution status based on certificates found by the **MOSEK** optimization library.

Solving a mixed-integer optimization program could easily result in long running time. It is therefore of interest to consider a termination criterion based on the maximum running time. This is possible setting the *MSK_DPAR_MIO_MAX_TIME*. See Section *4.8* for more details on how to set solver parameters.

4.6.2 Specifying an initial solution

Integer optimization problems are generally hard to solve, but the solution time can often be reduced by providing an initial solution for the solver. It is not necessary to specify the whole solution. By setting the *MSK_IPAR_MIO_CONSTRUCT_SOL* parameter to *MSK_ON* and inputting values for the integer variables only, will force **MOSEK** to compute the remaining continuous variable values.

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

Consider the problem

$$\begin{array}{ll} \text{maximize} & 7x_0 + 10x_1 + x_2 + 5x_3 \\ \text{subject to} & x_0 + x_1 + x_2 + x_3 \leq 2.5 \\ & x_0, x_1, x_2 \in \mathbb{Z} \\ x_0, x_1, x_2, x_3 \geq 0 & \end{array} \tag{4.8}$$

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

4.7 Solution Analysis

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

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

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

The first step in verifying that **MOSEK** reports the expected solution is to inspect the solution summary generated by **MOSEK** (see Section *4.7.1*). The solution summary provides information about

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

If the summary reports conflicting information (e.g. a solution status that does not match the actual solution), or the cause for terminating the solver before a solution was found cannot be traced back to the reasons stated above, it may be caused by a bug in the solver; in this case, please contact **MOSEK** support (see Section *1.2*).

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

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

and **MOSEK** reports the optimal solution

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

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

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

may have been inputted instead of

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

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

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

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

4.7.1 The Solution Summary

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

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

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

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

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

The interpreation of the solution summary is as follows:

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

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

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

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

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

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

Farkas' Lemma says

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

if and only if a y exists such that

$$\begin{array}{rcl} A^T y & \leq & 0, \\ b^T y & > & 0. \end{array} \tag{4.9}$$

Observe the infeasibility certificate has the same form as a regular dual solution and therefore the certificate is stored as a dual solution. In order to check quality of the primal infeasibility certificate it should be checked whether satisfies (4.9). Hence, the dual objective value is $b^T y$ should be strictly positive and the maximal violation in $A^T y \leq 0$ should be a small. In this case we conclude the certificate is of high quality because the dual objective is postive and large compared to the violations. Note the Farkas certificate is a ray so any postive multiple of that ray is also certificate. This implies the absolute of the value objective value and the violation is not relevant.

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

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

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

Listing 4.8 shows how to use these function to determine the quality of the solution.

Listing 4.8: An example of solution quality analysis.

```
lo1 <- list(sense="max", c=c(3,1,5,1))
lo1$A <- Matrix(c(3,1,2,0,2,1,3,1,0,2,0,3),
             nrow=3, byrow=TRUE, sparse=TRUE)
lo1$bc <- rbind(blc=c(30,15,-Inf), buc=c(30,Inf,25));
lo1$bx <- rbind(blx=c(0,0,0,0),    bux=c(Inf,10,Inf,Inf));

## Note we need to obtain details of the solution to inspect primal/dual obj fun value
r <- try(mosek(lo1, list(verbose=0, soldetail=2)), silent=TRUE)

if (inherits(r, "try-error")) {
  stop("Rmosek failed somehow!")
}

if (!identical(r$response$code, 0)) {
  cat(paste("**", "Response code:", r$response$code, "\n"))
  cat(paste("**", r$response$msg, "\n"))
  cat("Trying to continue..\n")
}

isdef <- try({
  itr <- r$sol$itr;
  itr$solsta;  itr$prosta;  itr$xx; itr$sol;
  }, silent=TRUE)

if (inherits(isdef, "try-error")) {
    stop("Interior solution was incomplete!")
}

if (itr$solsta == "OPTIMAL" |
    itr$solsta == "NEAR_OPTIMAL" ) {
```

```
  cat("The solution was (near) optimal, I am happy!\n")

  pobj<- r$sol$itr$pobjval
  dobj<- r$sol$itr$dobjval

  abs_obj_gap     <- abs(dobj-pobj)
  rel_obj_gap     <- abs_obj_gap/(1.0 + min(abs(pobj),abs(dobj)))


  max_primal_viol <- max( itr$maxinfeas$pcon, itr$maxinfeas$pbound)
  max_primal_viol <- max( max_primal_viol   , itr$maxinfeas$pbarvar)
  max_primal_viol <- max( max_primal_viol   , itr$maxinfeas$pcone)

  max_dual_viol <- max( itr$maxinfeas$dcon, itr$maxinfeas$dbound)
  max_dual_viol <- max( max_dual_viol     , itr$maxinfeas$dbarvar)
  max_dual_viol <- max( max_dual_viol     , itr$maxinfeas$dcone)


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

  cat("\n\n")
  cat("Customized solution information.\n")
  cat("  Absolute objective gap: ",abs_obj_gap,"\n")
  cat("  Relative objective gap: ",rel_obj_gap,"\n")
  cat("  Max primal violation  : ",max_primal_viol,"\n")
  cat("  Max dual violation    : ",max_dual_viol,"\n")

  accepted<- TRUE

  if (rel_obj_gap>1e-6){
    print ("Warning: The relative objective gap is LARGE.")
    accepted <- FALSE
  }

  ## We will accept a primal infeasibility of 1e-8 and
  ## dual infeasibility of 1e-6. These number should chosen problem
  ## dependent.

  if ( max_primal_viol>1e-8) {
    print ("Warning: Primal violation is too LARGE")
    accepted <- FALSE
  }

  if ( max_dual_viol>1e-6 ) {
    print ("Warning: Dual violation is too LARGE.")
    accepted <- FALSE
  }

  if ( accepted ) {
    numvar <- task.getnumvar()
    print ("Optimal primal solution")
    print(rbas$xx)
  }

} else {
    #Print detailed information about the solution
    cat(paste("**", "Solution status:", rbas$solsta, "\n"))
    cat(paste("**", "Problem status:", rbas$prosta, "\n"))
    stop("Solution could not be accepted!")
}
```

4.7.2 The Solution Summary for Mixed-Integer Problems

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

Listing 4.9: Example of solution summary for a mixed-integer problem.

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

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

4.8 Solver Parameters

The **MOSEK** API provides many parameters to tune and customize the solver behaviour. Parameters are grouped depending on their type: integer, double or string. In general, it should not be necessary to change any of the parameters but if required, it is easily done. A complete list of all parameters is found in Section *10.2*.

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

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

We need to set to *0* the parameter *MSK_IPAR_LOG*. Notice that in Rmosek the `MSK_IPAR_`, `MSK_DPAR_` and `MSK_SPAR_` prefixes for parameter names have been removed in favor of the `iparam`, `dparam` and `sparam` structures. Parameters are case-insensitive.

Assuming a problem named *lo1* has been defined, we can suppress solver verbosity as follow

```
lo1$iparam <- list(LOG = 0);
```

For more information about other parameter related functions, please browse the API reference in Section *10*.

Setting a parameter to `NULL` will remove it from the list according to the R language definition. Setting a parameter to `NA` or `NaN`, will on the other hand keep it on the list, only to be ignored by the interface with warnings confirming that this took place. Errors will be generated when a parameter name is not recognized or when the value defined for it is not within its feasible range.

CHAPTER

FIVE

NONLINEAR TUTORIALS

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

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

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

This leads to following advices in decreasing order of importance.

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

5.1 Separable Convex (SCopt) Interface

The **MOSEK** rmosek API provides a way to add simple non-linear functions composed from a limited set of non-linear terms. Non-linear terms can be mixed with quadratic terms in objective and constraints.

We consider a normal linear problem with additional non-linear terms z:

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

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

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

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

Table 5.1: Functions supported by the SCopt interface.

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

where f, g and h are constants.

This formulation does not guarantee convexity. For **MOSEK** to be able to solve the problem, following requirements must be met:

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

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

Important: When to use the SCopt API:

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

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

---

5.1.1 Ensuring Convexity and Differentiability

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

We can define these bounds as follows:

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

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

<table>
<tr><th>Separable function</th><th>Operator name</th><th>Convexity conditions</th></tr>
<tr><td>$fx\ln(x)$</td><td>ent</td><td>Convex within safe bounds.</td></tr>
<tr><td>fe^{gx+h}</td><td>exp</td><td>Convex for all x.</td></tr>
<tr><td>$f\ln(gx+h)$</td><td>log</td><td>Concave within safe bounds.</td></tr>
<tr><td rowspan="3">$f(x+h)^g$</td><td rowspan="3">pow</td><td>If g is even integer: convex within safe bounds.</td></tr>
<tr><td>If g is odd integer:
• concave if $(-\infty,-h)$,
• convex if $(-h,\infty)$
If $0 < g < 1$: concave within safe bounds.</td></tr>
<tr><td>Otherwise: convex within safe bounds.</td></tr>
</table>

5.1.2 SCopt Example

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

$$\begin{array}{ll} \text{maximize} & \exp(x_2) - \ln(x_1) \\ \text{subject to} & x_2 \ln(x_2) \geq 0 \\ & x_1^{\sqrt{2}} - x_2 \leq 0 \\ & x_1, x_2 \geq \frac{1}{2}. \end{array} \tag{5.1}$$

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

$$x_1, x_2 > 0.$$

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

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

From this example the input arguments for a separable convex program follow easily. The linear part of the objective function and constraints, as well as the constraint and variable bounds, should all be specified as for linear programs (see Section *4.2*.). The only addition to this is the list called *opro* containing the list-typed operator matrices (for objective) and (for constraints).

```
##                type  j     f     g     h
opro[,1] <- list("LOG", 1,  -1.0, 1.0, 0.0)
opro[,2] <- list("EXP", 2,   1.0, 1.0, 0.0)
```

The operator matrices have a column for each operator and a row for each descriptive element. The `opro` matrix have five rows called $\{type, j, f, g, h\}$, while the `oprc` matrix have six rows called $\{type, i, j, f, g, h\}$. Row should specify the operator type in a string, being either, exponential *EXP*, logarithm *LOG* or power *POW*. Row (not in *opro*) should specify the index of the constraint to which the non-linear term should be added. Row should specify the variable index of the operator. Rows , and should specify the coefficients of the operator.

```
##                type  i      j    f     g     h
oprc[,1] <- list("ENT", 1,     2, 1.0, 0.0, 0.0)
oprc[,2] <- list("POW", 2,     1, 1.0, 0.5, 0.0)
```

Note that the definition of the entropy operator, was the only operator defined without g and h. Thus, for entropy operators, these two unused rows in the operator matrix can be set to either zero or any empty definition (NULL, NA or NaN).

The complete source code follows in Listing 5.1.

Listing 5.1: Implementation of problem (5.1).

```
sco1 <- list(sense = "min")
sco1$c <- c(0,0)
sco1$A <- Matrix( c(0, 0,
                    0,-1), nrow=2, byrow=TRUE, sparse=TRUE)

sco1$bc <- rbind(blc = c(-Inf ,  0.),
                 buc = c(   0., Inf))
sco1$bx <- rbind(blx = rep(0.5, 2),
                 bux = rep(1.0, 2))

NUMOPRO <- 2; NUMOPRC <- 2;
opro <- matrix(list(), nrow=5, ncol=NUMOPRO)
oprc <- matrix(list(), nrow=6, ncol=NUMOPRC)
rownames(opro) <- c("type","j","f","g","h")
rownames(oprc) <- c("type","i","j","f","g","h")

##                type  j     f    g    h
opro[,1] <- list("LOG", 1,  -1.0, 1.0, 0.0)
opro[,2] <- list("EXP", 2,   1.0, 1.0, 0.0)

##                type  i     j    f    g    h
oprc[,1] <- list("ENT", 1,    2, 1.0, 0.0, 0.0)
oprc[,2] <- list("POW", 2,    1, 1.0, 0.5, 0.0)

sco1$scopt <- list(opro=opro, oprc=oprc)

r <- mosek(sco1)
```

CHAPTER

SIX

MANAGING I/O

The main purpose of this chapter is to give an overview on the logging and I/O features provided by the **MOSEK** package.

- Section *6.1* contains information about the log streams provided by **MOSEK**.
- File I/O is discussed in Section *6.2*.
- How to tune the logging verbosity is the topic of Section *6.3*.

6.1 Stream I/O

MOSEK execution produces a certain amount of loggging at environment and task level. This means that the logging from each environement and task can be isolated from the others.

The log messages are partitioned in three streams:

- **messages**
- **warnings**
- **errors**

These streams are aggregated in the **log** stream. See `MSKstreamtypee`.

6.2 File I/O

MOSEK supports a range of problem and solution formats listed in Section *11*. One such format is **MOSEK**'s native binary *Task format* which supports all features that **MOSEK** supports.

The file format used in I/O operations is deduced from extension - as in `problemname.task` - unless the parameter `MSK_IPAR_WRITE_DATA_FORMAT` is specified to something else. Problem files with an additional `.gz` extension - as in `problemname.task.gz` - are moreover assumed to use GZIP compression, and are automatically compressed, respectively decompressed, when written or read.

Example

If something is wrong with a problem or a solution, one option is to output the problem to the human-readable *OPF format* and inspect it by hand. For instance, one may use the `mosek_write` function to write the problem to a file immediately before optimizing it:

```
r <- mosek_write(lo1, "lo1.opf")
```

This will write the problem in `lo1` to the file `lo1.opf`. Similarly, using `mosek_read` is possible to read a problem from file. For instance the following code line

```
r <- mosek_read("lo1.opf")
```

will import a problem stored in the file `lo1.opf` in an object called *prob*.

6.3 Verbosity

The logging verbosity can be controlled by setting the relevant paramenters, as for instance

- *MSK_IPAR_LOG*,
- *MSK_IPAR_LOG_INTPNT*,
- *MSK_IPAR_LOG_MIO*,
- *MSK_IPAR_LOG_CUT_SECOND_OPT*,
- *MSK_IPAR_LOG_SIM*, and
- *MSK_IPAR_LOG_SIM_MINOR*.

Each parameter control the output level of a specific functionality or algorithm. The main switch is *MSK_IPAR_LOG* which affect the whole output. The actual log level for a specific functionality is determined as the minimum between *MSK_IPAR_LOG* and the relevant parameter. For instance, the log level for the output produce by the interior-point algorithm is tuned by the *MSK_IPAR_LOG_INTPNT*: the actual log level is defined by the minimum between *MSK_IPAR_LOG* and *MSK_IPAR_LOG_INTPNT*.

Tuning the solver verbosity may require adjusting several parameters. It must be noticed that verbose logging is supposed to be of interest during debugging and tuning, and it is consider the default setting. When output is no more of interest, user can easily disable using *MSK_IPAR_LOG*.

Moreover, it must be understood that larger values of *MSK_IPAR_LOG* do not necessarily result in an increased output.

By default **MOSEK** will reduce the amount of log information after the first optimization on a given task. To get full log output on subsequent optimizations set *MSK_IPAR_LOG_CUT_SECOND_OPT* to zero.

CHAPTER

SEVEN

PROBLEM FORMULATION AND SOLUTIONS

In this chapter we will discuss the following issues:

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

7.1 Linear Optimization

A linear optimization problem can be written as

$$\begin{array}{lccccc} \mbox{minimize} & & & c^T x + c^f & & \\ \mbox{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \end{array} \tag{7.1}$$

where

- m is the number of constraints.
- n is the number of decision variables.
- $x \in \mathbb{R}^n$ is a vector of decision variables.
- $c \in \mathbb{R}^n$ is the linear part of the objective function.
- $A \in \mathbb{R}^{m \times n}$ is the constraint matrix.
- $l^c \in \mathbb{R}^m$ is the lower limit on the activity for the constraints.
- $u^c \in \mathbb{R}^m$ is the upper limit on the activity for the constraints.
- $l^x \in \mathbb{R}^n$ is the lower limit on the activity for the variables.
- $u^x \in \mathbb{R}^n$ is the upper limit on the activity for the variables.

A primal solution (x) is *(primal) feasible* if it satisfies all constraints in (7.1). If (7.1) has at least one primal feasible solution, then (7.1) is said to be (primal) feasible.

In case (7.1) does not have a feasible solution, the problem is said to be *(primal) infeasible*

7.1.1 Duality for Linear Optimization

Corresponding to the primal problem (7.1), there is a dual problem

$$\begin{array}{ll} \mbox{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f \\ \mbox{subject to} & \begin{array}{rcl} A^T y + s_l^x - s_u^x & = & c, \\ -y + s_l^c - s_u^c & = & 0, \\ s_l^c, s_u^c, s_l^x, s_u^x & \geq & 0. \end{array} \end{array} \tag{7.2}$$

If a bound in the primal problem is plus or minus infinity, the corresponding dual variable is fixed at 0, and we use the convention that the product of the bound value and the corresponding dual variable is 0. E.g.

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

This is equivalent to removing variable $(s_l^x)_j$ from the dual problem. A solution

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

to the dual problem is feasible if it satisfies all the constraints in (7.2). If (7.2) has at least one feasible solution, then (7.2) is *(dual) feasible*, otherwise the problem is *(dual) infeasible*.

A Primal-dual Feasible Solution

A solution

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

is denoted a *primal-dual feasible solution*, if (x) is a solution to the primal problem (7.1) and $(y, s_l^c, s_u^c, s_l^x, s_u^x)$ is a solution to the corresponding dual problem (7.2).

The Duality Gap

Let

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

be a primal-dual feasible solution, and let

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

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

$$\begin{array}{l} c^T x^* + c^f - \left\{ (l^c)^T (s_l^c)^* - (u^c)^T (s_u^c)^* + (l^x)^T (s_l^x)^* - (u^x)^T (s_u^x)^* + c^f \right\} \\ \quad = \sum_{i=0}^{m-1} \left[(s_l^c)_i^* ((x_i^c)^* - l_i^c) + (s_u^c)_i^* (u_i^c - (x_i^c)^*) \right] \\ \quad + \sum_{j=0}^{n-1} \left[(s_l^x)_j^* (x_j - l_j^x) + (s_u^x)_j^* (u_j^x - x_j^*) \right] \geq 0 \end{array} \tag{7.3}$$

where the first relation can be obtained by transposing and multiplying the dual constraints (7.2) by x^* and $(x^c)^*$ respectively, and the second relation comes from the fact that each term in each sum is nonnegative. It follows that the primal objective will always be greater than or equal to the dual objective.

An Optimal Solution

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

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

are satisfied.

If (7.1) 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.

7.1.2 Infeasibility for Linear Optimization

Primal Infeasible Problems

If the problem (7.1) is infeasible (has no feasible solution), **MOSEK** will report a certificate of primal infeasibility: The dual solution reported is the certificate of infeasibility, and the primal solution is undefined.

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

$$\begin{array}{ll} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x \\ \text{subject to} & \\ & A^T y + s_l^x - s_u^x = 0, \\ & -y + s_l^c - s_u^c = 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0, \end{array} \tag{7.4}$$

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

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

to (7.4) so that

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

Such a solution implies that (7.4) is unbounded, and that its dual is infeasible. As the constraints to the dual of (7.4) are identical to the constraints of problem (7.1), we thus have that problem (7.1) is also infeasible.

Dual Infeasible Problems

If the problem (7.2) is infeasible (has no feasible solution), **MOSEK** will report a certificate of dual infeasibility: The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

$$\begin{array}{lccccc} \text{minimize} & & & c^T x & & \\ \text{subject to} & \hat{l}^c & \leq & Ax & \leq & \hat{u}^c, \\ & \hat{l}^x & \leq & x & \leq & \hat{u}^x, \end{array} \tag{7.5}$$

where

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

and

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

such that

$$c^T x < 0.$$

Such a solution implies that (7.5) is unbounded, and that its dual is infeasible. As the constraints to the dual of (7.5) are identical to the constraints of problem (7.2), we thus have that problem (7.2) is also infeasible.

Primal and Dual Infeasible Case

In case that both the primal problem (7.1) and the dual problem (7.2) are infeasible, **MOSEK** will report only one of the two possible certificates — which one is not defined (**MOSEK** returns the first certificate found).

Minimalization vs. Maximalization

When the objective sense of problem (7.1) is maximization, i.e.

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

the objective sense of the dual problem changes to minimization, and the domain of all dual variables changes sign in comparison to (7.2). The dual problem thus takes the form

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

This means that the duality gap, defined in (7.3) as the primal minus the dual objective value, becomes nonpositive. It follows that the dual objective will always be greater than or equal to the primal objective. The primal infeasibility certificate will be reported by **MOSEK** as a solution to the system

$$\begin{array}{r} A^T y + s_l^x - s_u^x = 0, \\ -y + s_l^c - s_u^c = 0, \\ s_l^c, s_u^c, s_l^x, s_u^x \leq 0, \end{array} \tag{7.6}$$

such that the objective value is strictly negative

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

Similarly, the certificate of dual infeasibility is an x satisfying the requirements of (7.5) such that $c^T x > 0$.

7.2 Conic Quadratic Optimization

Conic quadratic optimization is an extension of linear optimization (see Section *7.1*) allowing conic domains to be specified for subsets of the problem variables. A conic quadratic optimization problem can be written as

$$\begin{array}{lccccc} \text{minimize} & & & c^T x + c^f & & \\ \text{subject to} & l^c & \leq & Ax & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \\ & & & x \in \mathcal{K}, & & \end{array} \tag{7.7}$$

where set $\mathcal{K}$ is a Cartesian product of convex cones, namely $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_p$. Having the domain restriction, $x \in \mathcal{K}$, is thus equivalent to

$$x^t \in \mathcal{K}_t \subseteq \mathbb{R}^{n_t},$$

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

MOSEK supports only a limited number of cones, specifically:

- The $\mathbb{R}^n$ set.
- The quadratic cone:

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

- The rotated quadratic cone:

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

Although these cones may seem to provide only limited expressive power they can be used to model a wide range of problems as demonstrated in *[MOSEKApS12]*.

7.2.1 Duality for Conic Quadratic Optimization

The dual problem corresponding to the conic quadratic optimization problem (7.7) is given by

$$\begin{array}{ll}\text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f \\ \text{subject to} & \\ & A^T y + s_l^x - s_u^x + s_n^x = c \\ & -y + s_l^c - s_u^c = 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0, \\ & s_n^x \in \mathcal{K}^*, \end{array} \tag{7.8}$$

where the dual cone $\mathcal{K}^*$ is a Cartesian product of the cones

$$\mathcal{K}^* = \mathcal{K}_1^* \times \cdots \times \mathcal{K}_p^*,$$

where each $\mathcal{K}_t^*$ is the dual cone of $\mathcal{K}_t$. For the cone types **MOSEK** can handle, the relation between the primal and dual cone is given as follows:

- The $\mathbb{R}^n$ set:

$$\mathcal{K}_t = \mathbb{R}^{n_t} \quad \Leftrightarrow \quad \mathcal{K}_t^* = \{ s \in \mathbb{R}^{n_t} : \quad s = 0 \}.$$

- The quadratic cone:

$$\mathcal{K}_t = \mathcal{Q}^{n_t} \quad \Leftrightarrow \quad \mathcal{K}_t^* = \mathcal{Q}^{n_t} = \left\{ s \in \mathbb{R}^{n_t} : s_1 \geq \sqrt{\sum_{j=2}^{n_t} s_j^2} \right\}.$$

- The rotated quadratic cone:

$$\mathcal{K}_t = \mathcal{Q}_r^{n_t} \quad \Leftrightarrow \quad \mathcal{K}_t^* = \mathcal{Q}_r^{n_t} = \left\{ s \in \mathbb{R}^{n_t} : 2s_1s_2 \geq \sum_{j=3}^{n_t} s_j^2, \quad s_1 \geq 0, \quad s_2 \geq 0 \right\}.$$

Please note that the dual problem of the dual problem is identical to the original primal problem.

7.2.2 Infeasibility for Conic Quadratic Optimization

In case **MOSEK** finds a problem to be infeasible it reports a certificate of infeasibility. This works exactly as for linear problems (see Section *7.1.2*).

Primal Infeasible Problems

If the problem (7.7) is infeasible, **MOSEK** will report a certificate of primal infeasibility: The dual solution reported is the certificate of infeasibility, and the primal solution is undefined.

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

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

such that the objective value is strictly positive.

Dual infeasible problems

If the problem (7.8) is infeasible, **MOSEK** will report a certificate of dual infeasibility: The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

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

where

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

and

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

such that the objective value is strictly negative.

7.3 Semidefinite Optimization

Semidefinite optimization is an extension of conic quadratic optimization (see Section *7.2*) allowing positive semidefinite matrix variables to be used in addition to the usual scalar variables. A semidefinite optimization problem can be written as

$$\begin{array}{lccccll} \text{minimize} & & & \sum_{j=0}^{n-1} c_j x_j + \sum_{j=0}^{p-1} \langle \overline{C}_j, \overline{X}_j \rangle + c^f & & & \\ \text{subject to} & l_i^c & \leq & \sum_{j=0}^{n-1} a_{ij} x_j + \sum_{j=0}^{p-1} \langle \overline{A}_{ij}, \overline{X}_j \rangle & \leq & u_i^c, & i = 0, \ldots, m-1 \\ & l_j^x & \leq & x_j & \leq & u_j^x, & j = 0, \ldots, n-1 \\ & & & x \in \mathcal{K}, \overline{X}_j \in \mathcal{S}_+^{r_j}, & & & j = 0, \ldots, p-1 \end{array} \tag{7.9}$$

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

$$\langle U, V \rangle := \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} U_{ij} V_{ij}.$$

With semidefinite optimization we can model a wide range of problems as demonstrated in *[MOSEKApS12]*.

7.3.1 Duality for Semidefinite Optimization

The dual problem corresponding to the semidefinite optimization problem (7.9) is given by

$$\begin{array}{lll} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + c^f & \\ \text{subject to} & & \\ & c - A^T y + s_u^x - s_l^x = s_n^x, & \\ & \overline{C}_j - \sum_{i=0}^m y_i \overline{A}_{ij} = \overline{S}_j, & j = 0, \ldots, p-1 \\ & s_l^c - s_u^c = y, & \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0, & \\ & s_n^x \in \mathcal{K}^*, \quad \overline{S}_j \in \mathcal{S}_+^{r_j}, & j = 0, \ldots, p-1 \end{array} \tag{7.10}$$

where $A \in \mathbb{R}^{m\times n}$, $A_{ij} = a_{ij}$, which is similar to the dual problem for conic quadratic optimization (see Section *7.2.1*), except for the addition of dual constraints

$$\left(\overline{C}_j - \sum_{i=0}^{m} y_i \overline{A}_{ij}\right) \in \mathcal{S}_+^{r_j}.$$

Note that the dual of the dual problem is identical to the original primal problem.

7.3.2 Infeasibility for Semidefinite Optimization

In case **MOSEK** finds a problem to be infeasible it reports a certificate of the infeasibility. This works exactly as for linear problems (see Section *7.1.2*).

Primal Infeasible Problems

If the problem (7.9) is infeasible, **MOSEK** will report a certificate of primal infeasibility: The dual solution reported is a certificate of infeasibility, and the primal solution is undefined.

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

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

such that the objective value is strictly positive.

Dual Infeasible Problems

If the problem (7.10) is infeasible, **MOSEK** will report a certificate of dual infeasibility: The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

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

where

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

and

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

such that the objective value is strictly negative.

7.4 Quadratic and Quadratically Constrained Optimization

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

$$\begin{array}{lccccl} \text{minimize} & & & \frac{1}{2}x^T Q^o x + c^T x + c^f & & \\ \text{subject to} & l_k^c & \leq & \frac{1}{2}x^T Q^k x + \sum_{j=0}^{n-1} a_{kj} x_j & \leq & u_k^c, \quad k = 0, \ldots, m-1, \\ & l_j^x & \leq & x_j & \leq & u_j^x, \quad j = 0, \ldots, n-1, \end{array} \tag{7.11}$$

where Q^o and all Q^k are symmetric matrices. Moreover for convexity, Q^o must be a positive semidefinite matrix and Q^k must satisfy

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

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

7.4.1 A Recommendation

Any convex quadratic optimization problem can be reformulated as a conic quadratic optimization problem, see *[MOSEKApS12]* and in particular *[And13]*. In fact **MOSEK** does such conversion internally as a part of the solution process for the following reasons:

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

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

- it saves the computational overhead of the reformulation including the convexity check. A conic problem is convex by construction and hence no convexity check is neeeded for conic problems.
- usually the modeller can do a better reformulation than the automatic method because the modeller can exploit the knowledge of what is being modelled.

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

7.4.2 Duality for Quadratic and Quadratically Constrained Optimization

The dual problem corresponding to the quadratic and quadratically constrained optimization problem (7.11) is given by

$$\begin{array}{ll} \text{maximize} & (l^c)^T s_l^c - (u^c)^T s_u^c + (l^x)^T s_l^x - (u^x)^T s_u^x + \frac{1}{2}x^T \left\{ \sum_{k=0}^{m-1} y_k Q^k - Q^o \right\} x + c^f \\ \text{subject to} & \\ & A^T y + s_l^x - s_u^x + \left\{ \sum_{k=0}^{m-1} y_k Q^k - Q^o \right\} x = c, \\ & -y + s_l^c - s_u^c = 0, \\ & s_l^c, s_u^c, s_l^x, s_u^x \geq 0. \end{array} \tag{7.12}$$

The dual problem is related to the dual problem for linear optimization (see Section *7.1.1*), but depends on the variable x which in general can not be eliminated. In the solutions reported by **MOSEK**, the value of x is the same for the primal problem (7.11) and the dual problem (7.12).

7.4.3 Infeasibility for Quadratic and Quadratically Constrained Optimization

In case **MOSEK** finds a problem to be infeasible it reports a certificate of infeasibility. This works exactly as for linear problems (see Section *7.1.2*).

Primal Infeasible Problems

If the problem (7.11) with all $Q^k = 0$ is infeasible, **MOSEK** will report a certificate of primal infeasibility. As the constraints are the same as for a linear problem, the certificate of infeasibility is the same as for linear optimization (see Section *7.1.2.1*).

Dual Infeasible Problems

If the problem (7.12) with all $Q^k = 0$ is infeasible, **MOSEK** will report a certificate of dual infeasibility. The primal solution reported is the certificate of infeasibility, and the dual solution is undefined.

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

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

where

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

and

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

such that the objective value is strictly negative.

7.5 General Convex Optimization

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

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

where

- m is the number of constraints.
- n is the number of decision variables.
- $x \in \mathbb{R}^n$ is a vector of decision variables.
- $c \in \mathbb{R}^n$ is the linear part objective function.
- $A \in \mathbb{R}^{m \times n}$ is the constraint matrix.
- $l^c \in \mathbb{R}^m$ is the lower limit on the activity for the constraints.
- $u^c \in \mathbb{R}^m$ is the upper limit on the activity for the constraints.
- $l^x \in \mathbb{R}^n$ is the lower limit on the activity for the variables.

- $u^x \in \mathbb{R}^n$ is the upper limit on the activity for the variables.
- $f : \mathbb{R}^n \to \mathbb{R}$ is a nonlinear function.
- $g : \mathbb{R}^n \to \mathbb{R}^m$ is a nonlinear vector function.

This means that the i-th constraint has the form

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

The linear term Ax is included in $g(x)$ since it can be handled much more efficiently as a separate entity when optimizing.

The nonlinear functions f and g must be smooth in all $x \in [l^x; u^x]$. Moreover, $f(x)$ must be a convex function and $g_i(x)$ must satisfy

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

7.5.1 Duality for General convex Optimization

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

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

and the dual problem is given by

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

which is equivalent to

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

In this context we use the following definition for scalar functions

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

and accordingly for vector functions

$$\nabla g(x) = \left[\begin{array}{c} \nabla g_1(x) \\ \vdots \\ \nabla g_m(x) \end{array} \right].$$

CHAPTER

EIGHT

THE OPTIMIZERS FOR CONTINUOUS PROBLEMS

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

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

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

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

Using multiple threads

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

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

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

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

8.1 Presolve

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

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

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

After the presolved problem has been optimized the solution is automatically postsolved so that the returned solution is valid for the original problem. Hence, the presolve is completely transparent. For further details about the presolve phase, please see *[AA95]* and *[AGMX96]*.

It is possible to fine-tune the behavior of the presolve or to turn it off entirely. If presolve consumes too much time or memory compared to the reduction in problem size gained it may be disabled. This is done by setting the parameter *MSK_IPAR_PRESOLVE_USE* to *MSK_PRESOLVE_MODE_OFF*.

The two most time-consuming steps of the presolve are

- the eliminator, and
- the linear dependency check.

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

Numerical issues in the presolve

During the presolve the problem is reformulated so that it hopefully solves faster. However, in rare cases the presolved problem may be harder to solve then the original problem. The presolve may also be infeasible although the original problem is not.

If it is suspected that presolved problem is much harder to solve than the original then it is suggested to first turn the eliminator off by setting the parameter *MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES* to 0. If that does not help, then trying to turn presolve off may help.

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

Eliminator

The purpose of the eliminator is to eliminate free and implied free variables from the problem using substitution. For instance, given the constraints

$$\begin{array}{rcl} y & = & \sum_j x_j, \\ y, x & \geq & 0, \end{array}$$

y is an implied free variable that can be substituted out of the problem, if deemed worthwhile. If the eliminator consumes too much time or memory compared to the reduction in problem size gained it may be disabled. This can be done by setting the parameter *MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES* to 0. In rare cases the eliminator may cause that the problem becomes much hard to solve.

Linear dependency checker

The purpose of the linear dependency check is to remove linear dependencies among the linear equalities. For instance, the three linear equalities

$$\begin{array}{rcl} x_1 + x_2 + x_3 & = & 1, \\ x_1 + 0.5x_2 & = & 0.5, \\ 0.5x_2 + x_3 & = & 0.5 \end{array}$$

contain exactly one linear dependency. This implies that one of the constraints can be dropped without changing the set of feasible solutions. Removing linear dependencies is in general a good idea since it reduces the size of the problem. Moreover, the linear dependencies are likely to introduce numerical problems in the optimization phase.

It is best practise to build models without linear dependencies. If the linear dependencies are removed at the modeling stage, the linear dependency check can safely be disabled by setting the parameter *MSK_IPAR_PRESOLVE_LINDEP_USE* to *MSK_OFF*.

Dualizer

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

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

Note that currently only linear problems may be dualized.

Scaling

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

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

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

8.2 Linear Optimization

8.2.1 Optimizer Selection

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

8.2.2 The Interior-point Optimizer

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

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

$$\begin{array}{lccl} \text{minimize} & c^T x & & \\ \text{subject to} & Ax & = & b, \\ & x \geq 0. & & \end{array} \tag{8.1}$$

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

Since it is not known beforehand whether problem (8.1) has an optimal solution, is primal infeasible or is dual infeasible, the optimization algorithm must deal with all three situations. This is the reason that **MOSEK** solves the so-called homogeneous model

$$\begin{array}{rcl} Ax - b\tau & = & 0, \\ A^T y + s - c\tau & = & 0, \\ -c^T x + b^T y - \kappa & = & 0, \\ x, s, \tau, \kappa & \geq & 0, \end{array} \tag{8.2}$$

where y and s correspond to the dual variables in (8.1), and τ and κ are two additional scalar variables. Note that the homogeneous model (8.2) always has solution since

$$(x, y, s, \tau, \kappa) = (0, 0, 0, 0, 0)$$

is a solution, although not a very interesting one.

Any solution

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

to the homogeneous model (8.2) satisfies

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

Moreover, there is always a solution that has the property

$$\tau^* + \kappa^* > 0.$$

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

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

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

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

is a primal-dual optimal solution.

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

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

This implies that at least one of

$$-c^T x^* > 0 \tag{8.3}$$

or

$$b^T y^* > 0 \tag{8.4}$$

is satisfied. If (8.3) is satisfied then x^* is a certificate of dual infeasibility, whereas if (8.4) is satisfied then y^* is a certificate of dual infeasibility.

In summary, by computing an appropriate solution to the homogeneous model, all information required for a solution to the original problem is obtained. A solution to the homogeneous model can be computed using a primal-dual interior-point algorithm *[And09]*.

Interior-point Termination Criterion

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

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

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

to homogeneous model is generated where

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

Whenever the trial solution satisfies the criterion

$$\begin{aligned} \left\| A \frac{x^k}{\tau^k} - b \right\|_\infty &\leq \epsilon_p (1 + \|b\|_\infty), \\ \left\| A^T \frac{y^k}{\tau^k} + \frac{s^k}{\tau^k} - c \right\|_\infty &\leq \epsilon_d (1 + \|c\|_\infty), \text{ and} \\ \min\left(\frac{(x^k)^T s^k}{(\tau^k)^2}, \left| \frac{c^T x^k}{\tau^k} - \frac{b^T y^k}{\tau^k} \right| \right) &\leq \epsilon_g \max\left(1, \frac{\min(\left|c^T x^k\right|, \left|b^T y^k\right|)}{\tau^k}\right), \end{aligned} \tag{8.5}$$

the interior-point optimizer is terminated and

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

is reported as the primal-dual optimal solution. The interpretation of (8.5) is that the optimizer is terminated if

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

On the other hand, if the trial solution satisfies

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

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

$$\left\| A x^k \right\|_\infty > 0,$$

and define

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

It is easy to verify that

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

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

Finally, if

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

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

It is possible to adjust the tolerances ε_p, ε_d, ε_g and ε_i using parameters; see Table 8.1 for details.

Table 8.1: Parameters employed in termination criterion

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

The default values of the termination tolerances are chosen such that for a majority of problems appearing in practice it is not possible to achieve much better accuracy. Therefore, tightening the tolerances usually is not worthwhile. However, an inspection of (8.5) reveals that quality of the solution is dependent on $\|b\|_\infty$ and $\|c\|_\infty$; the smaller the norms are, the better the solution accuracy.

The interior-point method as implemented by **MOSEK** will converge toward optimality and primal and dual feasibility at the same rate *[And09]*. This means that if the optimizer is stopped prematurely then it is very unlikely that either the primal or dual solution is feasible. Another consequence is that in most cases all the tolerances, ε_p, ε_d and ε_g, have to be relaxed together to achieve an effect.

In some cases the interior-point method terminates having found a solution not too far from meeting the optimality condition (8.5). A solution is defined as *near optimal* if scaling ε_p, ε_d and ε_g by any number $\varepsilon_n \in [1.0, +\infty]$ conditions (8.5) are satisfied.

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

The basis identification discussed in Section *8.2.2.2* requires an optimal solution to work well; hence basis identification should be turned off if the termination criterion is relaxed.

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

Basis Identification

An interior-point optimizer does not return an optimal basic solution unless the problem has a unique primal and dual optimal solution. Therefore, the interior-point optimizer has an optional post-processing step that computes an optimal basic solution starting from the optimal interior-point solution. More information about the basis identification procedure may be found in *[AY96]*. In the following we provide an overall idea of the procedure.

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

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

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

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

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

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

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

In practice, when the algorithm gets close to the optimal solution, it is possible to construct in polynomial time an initial basis for the simplex algorithm from the current interior point solution. This basis is used to warm-start the simplex algorithm that will provide the optimal basic solution.

In most cases the constructed basis is optimal, or very few iterations are required by the simplex algorithm to make it optimal and hence the final *clean* phase be short. However, in some cases for nasty problems e.g. ill-conditioned problems the additional simplex clean up phase may take of lot a time.

By default **MOSEK** performs a basis identification. However, if a basic solution is not needed, the basis identification procedure can be turned off. The parameters

- *MSK_IPAR_INTPNT_BASIS*,
- *MSK_IPAR_BI_IGNORE_MAX_ITER*, and
- *MSK_IPAR_BI_IGNORE_NUM_ERROR*

control when basis identification is performed.

The type of simplex algorithm to be used can be tuned by the *MSK_IPAR_BI_CLEAN_OPTIMIZER* parameter i.e. primal or dual simplex, and the maximum number of iterations can be set by the *MSK_IPAR_BI_MAX_ITERATIONS*.

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

The Interior-point Log

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

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

The first line displays the number of threads used by the optimizer and second line tells that the optimizer chose to solve the dual problem rather than the primal problem. The next line displays the problem dimensions as seen by the optimizer, and the `Factor...` lines show various statistics. This is followed by the iteration log.

Using the same notation as in Section *8.2.2* the columns of the iteration log have the following meaning:

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

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

8.2.3 The simplex Based Optimizer

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

The simplex optimizer uses a different method that allows exploiting an initial guess for the optimal solution to reduce the solution time. Depending on the problem it may be faster or slower to use an initial guess; see section *8.2.4* for a discussion.

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

Simplex Termination Criterion

The simplex optimizer terminates when it finds an optimal basic solution or an infeasibility certificate. A basic solution is optimal when it is primal and dual feasible; see Section *7.1* and *7.1.1* for a definition of the primal and dual problem. Due to the fact that computations are performed in finite precision **MOSEK** allows violation of primal and dual feasibility within certain tolerances. The user can control the allowed primal and dual tolerances with the parameters *MSK_DPAR_BASIS_TOL_X* and *MSK_DPAR_BASIS_TOL_S* .

Starting From an Existing Solution

When using the simplex optimizer it may be possible to reuse an existing solution and thereby reduce the solution time significantly. When a simplex optimizer starts from an existing solution it is said to perform a *warm-start*. If the user is solving a sequence of optimization problems by solving the problem, making modifications, and solving again, **MOSEK** will warm-start automatically.

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

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

Numerical Difficulties in the Simplex Optimizers

Though **MOSEK** is designed to minimize numerical instability, completely avoiding it is impossible when working in finite precision. **MOSEK** counts a "numerical unexpected behavior" event inside the optimizer as a *set-back*. The user can define how many set-backs the optimizer accepts; if that number is exceeded, the optimization will be aborted. Set-backs are implemented to avoid long sequences where the optimizer tries to recover from an unstable situation.

Set-backs are, for example, repeated singularities when factorizing the basis matrix, repeated loss of feasibility, degeneracy problems (no progress in objective) and other events indicating numerical difficulties. If the simplex optimizer encounters a lot of set-backs the problem is usually badly scaled; in such a situation try to reformulate into a better scaled problem. Then, if a lot of set-backs still occur, trying one or more of the following suggestions may be worthwhile:

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

8.2.4 The Interior-point or the Simplex Optimizer?

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

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

8.2.5 The Primal or the Dual Simplex Variant?

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

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

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

8.3 Conic Optimization

8.3.1 The Interior-point Optimizer

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

Interior-point Termination Criteria

The parameters controlling when the conic interior-point optimizer terminates are shown in Table 8.2.

Table 8.2: Parameters employed in termination criterion.

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

8.4 Nonlinear Convex Optimization

8.4.1 The Interior-point Optimizer

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

The Convexity Requirement

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

The following parameters are available to control the convexity check:

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

The Differentiabilty Requirement

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

For instance, the function

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

is differentiable everywhere whereas the function

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

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

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

$$x \geq 0$$

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

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

$$f(x) = \frac{1}{x} + \frac{1}{1-x}. \tag{8.6}$$

In this case the bounds

$$0 \leq x \leq 1$$

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

For efficiency reasons it may be better to reformulate the problem than to force **MOSEK** to observe ranged bounds strictly. For instance, (8.6) can be reformulated as follows

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

Interior-point Termination Criteria

The parameters controlling when the general convex interior-point optimizer terminates are shown in Table 8.3.

Table 8.3: Parameters employed in termination criteria.

Parameter name	Purpose
MSK_DPAR_INTPNT_NL_TOL_PFEAS	Controls primal feasibility
MSK_DPAR_INTPNT_NL_TOL_DFEAS	Controls dual feasibility
MSK_DPAR_INTPNT_NL_TOL_REL_GAP	Controls relative gap
MSK_DPAR_INTPNT_TOL_INFEAS	Controls when the problem is declared infeasible
MSK_DPAR_INTPNT_NL_TOL_MU_RED	Controls when the complementarity is reduced enough

8.5 Using Multiple Threads in an Optimizer

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

8.5.1 Thread Safety

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

8.5.2 Determinism

The optimizers are run-to-run deterministic which means if a problem is solved twice on the same computer using the same parameter setting and exactly the same input then exactly the same results is obtained. One qualification is that no time limits must be imposed because the time taken to perform an operation on a computer is dependent on many factors such as the current workload.

8.5.3 The Parallelized Interior-point Optimizer

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

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

CHAPTER

NINE

THE OPTIMIZER FOR MIXED-INTEGER PROBLEMS

A problem is a mixed-integer optimization problem when one or more of the variables are constrained to be integer valued. **MOSEK** can solve mixed-integer

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

problems.

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

9.1 Some Concepts and Facts Related to Mixed-integer Optimization

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

In practice this implies that the focus should be on computing a near optimal solution quickly rather than on locating an optimal solution. Even if the problem is only solved approximately, it is important to know how far the approximate solution is from an optimal one. In order to say something about the quality of an approximate solution the concept of *relaxation* is important.

The mixed-integer optimization problem

$$\begin{array}{rcllcl} z^* & = & \text{minimize} & c^T x & & \\ & & \text{subject to} & Ax & = & b, \\ & & & x \geq 0 & & \\ & & & x_j \in \mathbb{Z}, & & \forall j \in \mathcal{J}, \end{array} \tag{9.1}$$

has the continuous relaxation

$$\begin{array}{rcllcl} \underline{z} & = & \text{minimize} & c^T x & & \\ & & \text{subject to} & Ax & = & b, \\ & & & x \geq 0 & & \end{array} \tag{9.2}$$

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

There are two important observations about the continuous relaxation. First, the continuous relaxation is usually much faster to optimize than the mixed-integer problem. Secondly if $\hat{x}$ is any feasible solution to (9.1) and

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

then

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

This is an important observation since if it is only possible to find a near optimal solution within a reasonable time frame then the quality of the solution can nevertheless be evaluated. The value $\underline{z}$ is a lower bound on the optimal objective value. This implies that the obtained solution is no further away from the optimum than $\bar{z} - \underline{z}$ in terms of the objective value.

Whenever a mixed-integer problem is solved **MOSEK** reports this lower bound so that the quality of the reported solution can be evaluated.

9.2 The Mixed-integer Optimizer

The mixed-integer optimizer can handle problems with linear, quadratic objective and constraints and conic constraints. However, a problem can not contain both quadratic objective or constraints and conic constraints.

The mixed-integer optimizer is specialized for solving linear and conic optimization problems. It can also solve pure quadratic and quadratically constrained problems; these problems are automatically converted to conic problems before being solved.

The mixed-integer optimizer is run-to-run deterministic. This means that if a problem is solved twice on the same computer with identical options then the obtained solution will be bit-for-bit identical for the two runs. However, if a time limit is set then this may not be case since the time taken to solve a problem is not deterministic. The mixed-integer optimizer is parallelized i.e. it can exploit multiple cores during the optimization.

The solution process can be split into these phases:

1. **Presolve:** In this phase the optimizer tries to reduce the size of the problem and improve the formulation using preprocessing techniques. The presolve stage can be turned off using the *MSK_IPAR_PRESOLVE_USE* parameter
2. **Cut generation:** Valid inequalities (cuts) are added to improve the lower bound
3. **Heuristic:** Using heuristics the optimizer tries to guess a good feasible solution. Heuristics can be controlled by the parameter *MSK_IPAR_MIO_HEURISTIC_LEVEL*
4. **Search:** The optimal solution is located by branching on integer variables

9.3 Termination Criterion

In general, it is time consuming to find an exact feasible and optimal solution to an integer optimization problem, though in many practical cases it may be possible to find a sufficiently good solution. Therefore, the mixed-integer optimizer employs a relaxed feasibility and optimality criterion to determine when a satisfactory solution is located.

A candidate solution that is feasible for the continuous relaxation is said to be an integer feasible solution if the criterion

$$\min(x_j - \lfloor x_j \rfloor, \lceil x_j \rceil - x_j) \leq \delta_1 \quad \forall j \in \mathcal{J}$$

is satisfied, meaning that x_j is at most δ_1 from the nearest integer.

Whenever the integer optimizer locates an integer feasible solution it will check if the criterion

$$\bar{z} - \underline{z} \leq \max(\delta_2, \delta_3 \max(10^{-10}, |\bar{z}|))$$

is satisfied. If this is the case, the integer optimizer terminates and reports the integer feasible solution as an optimal solution. 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.

9.3.1 Relaxed Termination

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

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

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

9.4 Parameters Affecting the Termination of the Integer Optimizer.

All δ tolerances can be adjusted using suitable parameters — see Table 9.1.

Table 9.1: Tolerances for the mixed-integer optimizer.

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

In Table 9.2 some other parameters affecting the integer optimizer termination criterion are shown. Please note that if the effect of a parameter is delayed, the associated termination criterion is applied only after some time, specified by the *MSK_DPAR_MIO_DISABLE_TERM_TIME* parameter.

Table 9.2: Other parameters affecting the integer optimizer termination criterion.

Parameter name	Delayed	Explanation
MSK_IPAR_MIO_MAX_NUM_BRANCHES	Yes	Maximum number of branches allowed.
MSK_IPAR_MIO_MAX_NUM_RELAXS	Yes	Maximum number of relaxations allowed.
MSK_IPAR_MIO_MAX_NUM_SOLUTIONS	Yes	Maximum number of feasible integer solutions allowed.

9.5 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 *9.3* for details.
- Specify a good initial solution: In many cases a good feasible solution is either known or easily computed using problem specific knowledge. If a good feasible solution is known, it is usually worthwhile to use this as a starting point for the integer optimizer.
- Improve the formulation: A mixed-integer optimization problem may be impossible to solve in one form and quite easy in another form. However, it is beyond the scope of this manual to discuss good formulations for mixed-integer problems. For discussions on this topic see for example *[Wol98]*.

9.6 Understanding Solution Quality

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

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

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

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

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

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

CHAPTER

TEN

API REFERENCE

10.1 Command Reference

The Rmosek interface is composed by a small set of functions and structures.

- *Function list*
- *Structures and data types list*

10.1.1 Functions

- `mosek`
- `mosek_version`
- `mosek_clean`
- `mosek_read`
- `mosek_write`

`r = mosek(prob, opts)`

Solve an optimization problem using the **MOSEK** optimization library.

`Parameters`

- `[in]` `prob` (*problem*) – a struct describing the optimization problem.
- `[in]` `opts` (*options*) – the solver options

`Return`

- `r` (*result*) – a struct that contains the result of the optimization

`ver = mosek_version()`

Retrieves a string containing the version number of the utilized **MOSEK** optimization library.

`Return`

- `ver` (`int`) – The version number.

`mosek_clean()`

Forces the early release of any previously acquired **MOSEK** license. If you do not share a limited number of licenses among multiple users, you do not need to use this function. The acquisition of a new **MOSEK** license will automatically take place at the next call to the function `mosek` given a valid problem description, using a small amount of extra time.

For advanced users: If you utilize the *.Call* convention directly, i.e. bypassing the mosek R-function definition, an `Rf_error` will result in an unclean memory space. For this reason you can also use this function to tidy up uncleaned resources in case an error occurs. Otherwise this cleaning will not happen until the next call to `mosek` or until the library is unloaded. This usage have not been documented elsewhere.

out = mosek_read(filepath, opts = list())

Interprets a model from any standard modeling fileformat (e.g. lp, opf, mps, task, etc.), controlled by a set of options. The result contains an optimization problem which is compliant with the input specifications of function *mosek*.

Parameters

- [in] filepath (string) – A string describing the path to file, either absolute or relative to the working directory. The specified location will be the source of the optimization model to be read.
- [in] opts (*io_options*) – The options could have any name, and are, in fact, often input directly as an anonymous list.

Return

- out (*result*) – The resulting function output variable , returned by the interface, holds the response of the function call.

r = mosek_write(prob, filepath, opts)

Outputs a model of an optimization problem in a file format (see Section *11*), controlled by a set of options. The modeling file format is selected based on the extension of the modelfile.

Parameters

- [in] prob (*problem*) – The input variable could have any name, but should be a list object describing the optimization problem using the same fields as for the *mosek* function.
- [in] filepath (string) – The input variable should be a string describing the path to modelfile. This path can either be absolute or relative to the working directory, and will overwrite any existing data on this location. The specified location will be the destination of the exported model.
- [in] opts (*io_options*) – The options could have any name, and are, in fact, often specified directly as an anonymous list.

Return

- r (*result*) – Function return information.

10.1.2 Structures and Data Types

rescode

The return code type. See Section *10.3*.

problem

A list object describing the optimization problem using the following fields.

Fields

- sense (string) – Objective sense, e.g. "max" or "min"
- c (numeric_vector) – Objective coefficient array.
- c0 (numeric) – Objective constant.
- A (matrix_sparse) – Constraint sparse matrix.
- bc (matrix) – Lower and upper constraint bounds
- bx (matrix) – Lower and upper variable bounds
- cones (matrix_list) – Conic constraints
- bardim (numeric_vector) – Semidefinite variable dimensions
- barc (list) – Semidefinite objective coefficients
- barA (list) – Semidefinite constraint coefficients
- intsub (numeric_vector) – Integer variable indexes
- qobj (list) – [*optional*] Quadratic convex optimization

- `scopt` (`list`) – [*optional*] Separable convex optimization
- `iparam` (`list`) – Integer parameter list
- `dparam` (`list`) – Double parameter list
- `sparam` (`list`) – String parameter list
- `sol` (*solver_solutions*) – Initial solution struct

`options`

The options could have any name, and are, in fact, often input directly as an anonymous list.

`Fields`

- `verbose` (`numeric`) – Output logging verbosity
- `usesol` (`bool`) – Whether to use the initial solution
- `useparam` (`bool`) – Whether to use the specified parameter settings
- `soldetail` (`numeric`) – Level of detail used to describe solutions
- `getinfo` (`bool`) – Whether to extract **MOSEK** information items
- `writebefore` (`string`) – Filepath used to export model
- `writeafter` (`string`) – Filepath used to export model and solution

`solver_solutions`

It contains informations about initial/final solutions.

> **Warning:** Fields availability depends on the type of problem/algorithm.

`Fields`

- `itr` (*solution_info*) – Interior solution
- `bas` (*solution_info*) – Basic solution
- `int` (*solution_info*) – Integer solution

`solution_info`

`Fields`

- `solsta` (`string`) – Solution status
- `prosta` (`string`) – Problem status
- `skc` (`string_vector`) – Linear constraint status keys
- `skx` (`string_vector`) – Variable bound status keys
- `skn` (`string_vector`) – Conic constraint status keys (not in basic solution)
- `xc` (`numeric_vector`) – Constraint activities
- `xx` (`numeric_vector`) – Variable activities
- `barx` (`numeric_vector`) – Semidefinite variable activities (not in basic solution)
- `slc` (`numeric_vector`) – Dual variable for constraint lower bounds (not in integer solution)
- `suc` (`numeric_vector`) – Dual variable for constraint upper bounds (not in integer solution)
- `slx` (`numeric_vector`) – Dual variable for variable lower bounds (not in integer solution)
- `sux` (`numeric_vector`) – Dual variable for variable lower bounds (not in integer solution)
- `snx` (`numeric_vector`) – Dual variable of conic constraints (not in basic or integer solution)
- `bars` (`numeric_vector`) – Dual variable of semidefinite domains (not in basic or integer solution)

- pobjval (numeric) – Primal objective value (Only available if requested by option soldetail)
- dobjval (numeric) – Dual objective value (not fo integer solution)
- pobjbound (numeric) – Best primal objective bound from relaxations (only for integer solution)
- maxinfeas (*infeas_info*) – Maximal solution infeasibilities

infeas_info
: It contains information about the maximal solution infeasibilty for several problem items.

Note: This struct is only available if the option soldetail is specified.

Fields

- pbound (numeric) – In primal inequality constraints
- peq (numeric) – In primal equality constraints
- pcone (numeric) – In primal cone constraints
- dbound (numeric) – In dual inequality constraints
- deq (numeric) – In dual equality constraints
- dcone (numeric) – In dual cone constraints
- int (numeric) – In integer variables

io_options
: It is used to specify options for input/output operations.

Fields

- verbose (numeric) – Output logging verbosity
- usesol (bool) – Whether to write an initial solution
- useparam (bool) – Whether to write all parameter settings
- getinfo (bool) – Whether to extract **MOSEK** information items
- scofile (string) – Source of operators read to scopt
- matrixformat (matrix_sparse) – The sparse format of the constraint matrix (only used by *mosek_read*).

result
: It contains results for most of the Rmosek functions.

Note: Some fields are available only for specific functions.

Fields

- response (*rescode*) – Response from the **MOSEK** optimization library.
- code (numeric) – ID-code of response
- msg (string) – Human-readable message
- prob (*problem*) – Problem description (only available in *mosek_read*)
- sol (*solution_info*) – Available solutions (algorithm/problem dependent) (only available in *mosek*)
- iinfo (numeric_list) – Integer information list (Only available if requested by option getinfo).
- dinfo (numeric_list) – Double information list (Only available if requested by option getinfo).

10.2 Parameters

All parameters (alphabetical order)

- *double parameters*
- *integer parameters*
- *string parameters*

Parameters grouped by topic

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

- *Logging*
- *Basis identification*
- *Output information*
- *Infeasibility report*
- *Data check*
- *Conic interior-point method*
- *Primal simplex optimizer*
- *Nonlinear convex method*
- *Optimization system*
- *Progress call-back*
- *Overall solver*
- *Simplex optimizer*
- *Solution input/output*
- *Debugging*
- *Termination criterion*
- *License manager*
- *Dual simplex optimizer*
- *Data input/output*
- *Presolve*
- *Interior-point method*
- *Analysis*
- *Mixed-integer optimization*

10.2.1 Parameters List (alphabetically)

Double Parameters

`MSK_DPAR_ANA_SOL_INFEAS_TOL`

If a constraint violates its bound with an amount larger than this value, the constraint name, index and violation will be printed by the solution analyzer.

`Accepted Values:` [0.0 ;+inf]

`Default Value:` 1e-6

Groups: *Analysis*

MSK_DPAR_BASIS_REL_TOL_S

Maximum relative dual bound violation allowed in an optimal basic solution.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e-12

Groups: *Simplex optimizer*, *Termination criterion*

MSK_DPAR_BASIS_TOL_S

Maximum absolute dual bound violation in an optimal basic solution.

Accepted Values: [1.0e-9 ;+inf]

Default Value: 1.0e-6

Groups: *Simplex optimizer*, *Termination criterion*

MSK_DPAR_BASIS_TOL_X

Maximum absolute primal bound violation allowed in an optimal basic solution.

Accepted Values: [1.0e-9 ;+inf]

Default Value: 1.0e-6

Groups: *Simplex optimizer*, *Termination criterion*

MSK_DPAR_CHECK_CONVEXITY_REL_TOL

This parameter controls when the full convexity check declares a problem to be non-convex. Increasing this tolerance relaxes the criteria for declaring the problem non-convex.

A problem is declared non-convex if negative (positive) pivot elements are detected in the Cholesky factor of a matrix which is required to be PSD (NSD). This parameter controls how much this non-negativity requirement may be violated.

If d_i is the pivot element for column i, then the matrix Q is considered to not be PSD if:

$$d_i \leq -|Q_{ii}|\texttt{check\_convexity\_rel\_tol}$$

Accepted Values: [0 ;+inf]

Default Value: 1e-10

Groups: *Interior-point method*

MSK_DPAR_DATA_SYM_MAT_TOL

Absolute zero tolerance for elements in in suymmetric matrixes. If any value in a symmetric matrix is smaller than this parameter in absolute terms **MOSEK** will treat the values as zero and generate a warning.

Accepted Values: [1.0e-16 ;1.0e-6]

Default Value: 1.0e-12

Groups: *Data check*

MSK_DPAR_DATA_SYM_MAT_TOL_HUGE

An element in a symmetric matrix which is larger than this value in absolute size causes an error.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e20

Groups: *Data check*

MSK_DPAR_DATA_SYM_MAT_TOL_LARGE

An element in a symmetric matrix which is larger than this value in absolute size causes a warning message to be printed.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e10

Groups: *Data check*

MSK_DPAR_DATA_TOL_AIJ

Absolute zero tolerance for elements in A. If any value A_{ij} is smaller than this parameter in absolute terms **MOSEK** will treat the values as zero and generate a warning.

Accepted Values: [1.0e-16 ;1.0e-6]

Default Value: 1.0e-12

Groups: *Data check*

MSK_DPAR_DATA_TOL_AIJ_HUGE

An element in A which is larger than this value in absolute size causes an error.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e20

Groups: *Data check*

MSK_DPAR_DATA_TOL_AIJ_LARGE

An element in A which is larger than this value in absolute size causes a warning message to be printed.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e10

Groups: *Data check*

MSK_DPAR_DATA_TOL_BOUND_INF

Any bound which in absolute value is greater than this parameter is considered infinite.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e16

Groups: *Data check*

MSK_DPAR_DATA_TOL_BOUND_WRN

If a bound value is larger than this value in absolute size, then a warning message is issued.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e8

Groups: *Data check*

MSK_DPAR_DATA_TOL_CJ_LARGE

An element in c which is larger than this value in absolute terms causes a warning message to be printed.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e8

Groups: *Data check*

MSK_DPAR_DATA_TOL_C_HUGE

An element in c which is larger than the value of this parameter in absolute terms is considered to be huge and generates an error.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e16

Groups: *Data check*

`MSK_DPAR_DATA_TOL_QIJ`
Absolute zero tolerance for elements in Q matrices.

`Accepted Values:` [0.0 ;+inf]

`Default Value:` 1.0e-16

`Groups:` *Data check*

`MSK_DPAR_DATA_TOL_X`
Zero tolerance for constraints and variables i.e. if the distance between the lower and upper bound is less than this value, then the lower and upper bound is considered identical.

`Accepted Values:` [0.0 ;+inf]

`Default Value:` 1.0e-8

`Groups:` *Data check*

`MSK_DPAR_INTPNT_CO_TOL_DFEAS`
Dual feasibility tolerance used by the conic interior-point optimizer.

`Accepted Values:` [0.0 ;1.0]

`Default Value:` 1.0e-8

`Groups:` *Interior-point method*, *Termination criterion*, *Conic interior-point method*

`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.

`Accepted Values:` [0.0 ;1.0]

`Default Value:` 1.0e-10

`Groups:` *Interior-point method*, *Termination criterion*, *Conic interior-point method*

`MSK_DPAR_INTPNT_CO_TOL_MU_RED`
Relative complementarity gap feasibility tolerance used by the conic interior-point optimizer.

`Accepted Values:` [0.0 ;1.0]

`Default Value:` 1.0e-8

`Groups:` *Interior-point method*, *Termination criterion*, *Conic interior-point method*

`MSK_DPAR_INTPNT_CO_TOL_NEAR_REL`
If **MOSEK** cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.

`Accepted Values:` [1.0 ;+inf]

`Default Value:` 1000

`Groups:` *Interior-point method*, *Termination criterion*, *Conic interior-point method*

`MSK_DPAR_INTPNT_CO_TOL_PFEAS`
Primal feasibility tolerance used by the conic interior-point optimizer.

`Accepted Values:` [0.0 ;1.0]

`Default Value:` 1.0e-8

`Groups:` *Interior-point method*, *Termination criterion*, *Conic interior-point method*

`MSK_DPAR_INTPNT_CO_TOL_REL_GAP`
Relative gap termination tolerance used by the conic interior-point optimizer.

`Accepted Values:` [0.0 ;1.0]

`Default Value:` 1.0e-7

Groups: *Interior-point method*, *Termination criterion*, *Conic interior-point method*

MSK_DPAR_INTPNT_NL_MERIT_BAL
Controls if the complementarity and infeasibility is converging to zero at about equal rates.

Accepted Values: [0.0 ;0.99]

Default Value: 1.0e-4

Groups: *Interior-point method*, *Nonlinear convex method*

MSK_DPAR_INTPNT_NL_TOL_DFEAS
Dual feasibility tolerance used when a nonlinear model is solved.

Accepted Values: [0.0 ;1.0]

Default Value: 1.0e-8

Groups: *Interior-point method*, *Termination criterion*, *Nonlinear convex method*

MSK_DPAR_INTPNT_NL_TOL_MU_RED
Relative complementarity gap tolerance.

Accepted Values: [0.0 ;1.0]

Default Value: 1.0e-12

Groups: *Interior-point method*, *Termination criterion*, *Nonlinear convex method*

MSK_DPAR_INTPNT_NL_TOL_NEAR_REL
If the **MOSEK** nonlinear interior-point optimizer cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.

Accepted Values: [1.0 ;+inf]

Default Value: 1000.0

Groups: *Interior-point method*, *Termination criterion*, *Nonlinear convex method*

MSK_DPAR_INTPNT_NL_TOL_PFEAS
Primal feasibility tolerance used when a nonlinear model is solved.

Accepted Values: [0.0 ;1.0]

Default Value: 1.0e-8

Groups: *Interior-point method*, *Termination criterion*, *Nonlinear convex method*

MSK_DPAR_INTPNT_NL_TOL_REL_GAP
Relative gap termination tolerance for nonlinear problems.

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

Default Value: 1.0e-6

Groups: *Termination criterion*, *Interior-point method*, *Nonlinear convex method*

MSK_DPAR_INTPNT_NL_TOL_REL_STEP
Relative step size to the boundary for general nonlinear optimization problems.

Accepted Values: [1.0e-4 ;0.9999999]

Default Value: 0.995

Groups: *Interior-point method*, *Nonlinear convex method*

MSK_DPAR_INTPNT_QO_TOL_DFEAS
Dual feasibility tolerance used when the interior-point optimizer is applied to a quadratic optimization problem..

Accepted Values: [0.0 ;1.0]

Default Value: 1.0e-8

Groups: *Interior-point method*, *Termination criterion*

MSK_DPAR_INTPNT_QO_TOL_INFEAS

Controls when the conic interior-point optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

Accepted Values: [0.0 ;1.0]

Default Value: 1.0e-10

Groups: *Interior-point method*, *Termination criterion*

MSK_DPAR_INTPNT_QO_TOL_MU_RED

Relative complementarity gap feasibility tolerance used when interior-point optimizer is applied to a quadratic optimization problem.

Accepted Values: [0.0 ;1.0]

Default Value: 1.0e-8

Groups: *Interior-point method*, *Termination criterion*

MSK_DPAR_INTPNT_QO_TOL_NEAR_REL

If **MOSEK** cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.

Accepted Values: [1.0 ;+inf]

Default Value: 1000

Groups: *Interior-point method*, *Termination criterion*

MSK_DPAR_INTPNT_QO_TOL_PFEAS

Primal feasibility tolerance used when the interior-point optimizer is applied to a quadratic optimization problem.

Accepted Values: [0.0 ;1.0]

Default Value: 1.0e-8

Groups: *Interior-point method*, *Termination criterion*

MSK_DPAR_INTPNT_QO_TOL_REL_GAP

Relative gap termination tolerance used when the interior-point optimizer is applied to a quadratic optimization problem.

Accepted Values: [0.0 ;1.0]

Default Value: 1.0e-8

Groups: *Interior-point method*, *Termination criterion*

MSK_DPAR_INTPNT_TOL_DFEAS

Dual feasibility tolerance used for linear and quadratic optimization problems.

Accepted Values: [0.0 ;1.0]

Default Value: 1.0e-8

Groups: *Interior-point method*, *Termination criterion*

MSK_DPAR_INTPNT_TOL_DSAFE

Controls the initial dual starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly and/or the constraint or variable bounds are very large, then it might be worthwhile to increase this value.

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

Default Value: 1.0

Groups: *Interior-point method*

MSK_DPAR_INTPNT_TOL_INFEAS

Controls when the optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible. A value of 0.0 means the optimizer must have an exact certificate of infeasibility and this is very unlikely to happen.

Accepted Values: [0.0 ;1.0]

Default Value: 1.0e-10

Groups: *Interior-point method*, *Termination criterion*, *Nonlinear convex method*

MSK_DPAR_INTPNT_TOL_MU_RED

Relative complementarity gap tolerance.

Accepted Values: [0.0 ;1.0]

Default Value: 1.0e-16

Groups: *Interior-point method*, *Termination criterion*

MSK_DPAR_INTPNT_TOL_PATH

Controls how close the interior-point optimizer follows the central path. A large value of this parameter means the central is followed very closely. On numerical unstable problems it may be worthwhile to increase this parameter.

Accepted Values: [0.0 ;0.9999]

Default Value: 1.0e-8

Groups: *Interior-point method*

MSK_DPAR_INTPNT_TOL_PFEAS

Primal feasibility tolerance used for linear and quadratic optimization problems.

Accepted Values: [0.0 ;1.0]

Default Value: 1.0e-8

Groups: *Interior-point method*, *Termination criterion*

MSK_DPAR_INTPNT_TOL_PSAFE

Controls the initial primal starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly and/or the constraint or variable bounds are very large, then it may be worthwhile to increase this value.

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

Default Value: 1.0

Groups: *Interior-point method*

MSK_DPAR_INTPNT_TOL_REL_GAP

Relative gap termination tolerance.

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

Default Value: 1.0e-8

Groups: *Termination criterion*, *Interior-point method*

MSK_DPAR_INTPNT_TOL_REL_STEP

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

Accepted Values: [1.0e-4 ;0.999999]

Default Value: 0.9999

Groups: *Interior-point method*

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

Accepted Values: [0.0 ;1.0]

Default Value: 1.0e-6

Groups: *Interior-point method*

MSK_DPAR_LOWER_OBJ_CUT
If either a primal or dual feasible solution is found proving that the optimal objective value is outside, the interval [*MSK_DPAR_LOWER_OBJ_CUT*, *MSK_DPAR_UPPER_OBJ_CUT*], then **MOSEK** is terminated.

Accepted Values: [-inf ;+inf]

Default Value: -1.0e30

Groups: *Termination criterion*

MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH
If the lower objective cut is less than the value of this parameter value, then the lower objective cut i.e. *MSK_DPAR_LOWER_OBJ_CUT* is treated as $-\infty$.

Accepted Values: [-inf ;+inf]

Default Value: -0.5e30

Groups: *Termination criterion*

MSK_DPAR_MIO_DISABLE_TERM_TIME
This parameter specifies the number of seconds n during which the termination criteria governed by

- *MSK_IPAR_MIO_MAX_NUM_RELAXS*
- *MSK_IPAR_MIO_MAX_NUM_BRANCHES*
- *MSK_DPAR_MIO_NEAR_TOL_ABS_GAP*
- *MSK_DPAR_MIO_NEAR_TOL_REL_GAP*

is disabled since the beginning of the optimization.

A negative value is identical to infinity i.e. the termination criteria are never checked.

Accepted Values: [-inf ;+inf]

Default Value: -1.0

Groups: *Mixed-integer optimization*, *Termination criterion*

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

Accepted Values: [-inf ;+inf]

Default Value: -1.0

Groups: *Mixed-integer optimization*, *Termination criterion*

MSK_DPAR_MIO_NEAR_TOL_ABS_GAP
Relaxed absolute optimality tolerance employed by the mixed-integer optimizer. This termination criteria is delayed. See *MSK_DPAR_MIO_DISABLE_TERM_TIME* for details.

Accepted Values: [0.0 ;+inf]

Default Value: 0.0

Groups: *Mixed-integer optimization*

MSK_DPAR_MIO_NEAR_TOL_REL_GAP
The mixed-integer optimizer is terminated when this tolerance is satisfied. This termination criteria is delayed. See *MSK_DPAR_MIO_DISABLE_TERM_TIME* for details.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e-3

Groups: *Mixed-integer optimization*, *Termination criterion*

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

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

Default Value: 1.0e-10

Groups: *Mixed-integer optimization*, *Termination criterion*

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

Accepted Values: [0.0 ;+inf]

Default Value: 0.0

Groups: *Mixed-integer optimization*

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

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

Default Value: 1.0e-5

Groups: *Mixed-integer optimization*

MSK_DPAR_MIO_TOL_FEAS
Feasibility tolerance for mixed integer solver.

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

Default Value: 1.0e-6

Groups: *Mixed-integer optimization*

MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT
If the relative improvement of the dual bound is smaller than this value, the solver will terminate the root cut generation. A value of 0.0 means that the value is selected automatically.

Accepted Values: [0.0 ;1.0]

Default Value: 0.0

Groups: *Mixed-integer optimization*

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

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e-4

Groups: *Mixed-integer optimization*, *Termination criterion*

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

Accepted Values: [-inf ;+inf]

Default Value: -1.0

Groups: *Termination criterion*

MSK_DPAR_PRESOLVE_TOL_ABS_LINDEP
Absolute tolerance employed by the linear dependency checker.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e-6

Groups: *Presolve*

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

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

Default Value: 1.0e-12

Groups: *Presolve*

MSK_DPAR_PRESOLVE_TOL_REL_LINDEP
Relative tolerance employed by the linear dependency checker.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e-10

Groups: *Presolve*

MSK_DPAR_PRESOLVE_TOL_S
Absolute zero tolerance employed for s_i in the presolve.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e-8

Groups: *Presolve*

MSK_DPAR_PRESOLVE_TOL_X
Absolute zero tolerance employed for x_j in the presolve.

Accepted Values: [0.0 ;+inf]

Default Value: 1.0e-8

Groups: *Presolve*

MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL
This parameter determines when columns are dropped in incomplete Cholesky factorization during reformulation of quadratic problems.

Accepted Values: [0 ;+inf]

Default Value: 1e-15

Groups: *Interior-point method*

MSK_DPAR_SEMIDEFINITE_TOL_APPROX
Tolerance to define a matrix to be positive semidefinite.

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

Default Value: 1.0e-10

Groups: *Data check*

MSK_DPAR_SIMPLEX_ABS_TOL_PIV
Absolute pivot tolerance employed by the simplex optimizers.

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

Default Value: 1.0e-7

Groups: *Simplex optimizer*

`MSK_DPAR_SIM_LU_TOL_REL_PIV`
Relative pivot tolerance employed when computing the LU factorization of the basis in the simplex optimizers and in the basis identification procedure.

A value closer to 1.0 generally improves numerical stability but typically also implies an increase in the computational work.

`Accepted Values:` [1.0e-6 ;0.999999]

`Default Value:` 0.01

`Groups:` *Basis identification*, *Simplex optimizer*

`MSK_DPAR_UPPER_OBJ_CUT`
If either a primal or dual feasible solution is found proving that the optimal objective value is outside, the interval [*MSK_DPAR_LOWER_OBJ_CUT*, *MSK_DPAR_UPPER_OBJ_CUT*], then **MOSEK** is terminated.

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

`Default Value:` 1.0e30

`Groups:` *Termination criterion*

`MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH`
If the upper objective cut is greater than the value of this parameter, then the upper objective cut *MSK_DPAR_UPPER_OBJ_CUT* is treated as ∞.

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

`Default Value:` 0.5e30

`Groups:` *Termination criterion*

Integer Parameters

`MSK_IPAR_ANA_SOL_BASIS`
Controls whether the basis matrix is analyzed in solution analyzer.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_ON*

`Groups:` *Analysis*

`MSK_IPAR_ANA_SOL_PRINT_VIOLATED`
Controls whether a list of violated constraints is printed.

All constraints violated by more than the value set by the parameter *MSK_DPAR_ANA_SOL_INFEAS_TOL* will be printed.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_OFF*

`Groups:` *Analysis*

`MSK_IPAR_AUTO_SORT_A_BEFORE_OPT`
Controls whether the elements in each column of A are sorted before an optimization is performed. This is not required but makes the optimization more deterministic.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_OFF*

`Groups:` *Debugging*

MSK_IPAR_AUTO_UPDATE_SOL_INFO

Controls whether the solution information items are automatically updated after an optimization is performed.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Optimization system*

MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE

If a slack variable is in the basis, then the corresponding column in the basis is a unit vector with -1 in the right position. However, if this parameter is set to *MSK_ON*, -1 is replaced by 1.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Simplex optimizer*

MSK_IPAR_BI_CLEAN_OPTIMIZER

Controls which simplex optimizer is used in the clean-up phase.

Accepted Values: *MSKoptimizertypee*

Default Value: *MSK_OPTIMIZER_FREE*

Groups: *Basis identification*, *Overall solver*

MSK_IPAR_BI_IGNORE_MAX_ITER

If the parameter *MSK_IPAR_INTPNT_BASIS* has the value *MSK_BI_NO_ERROR* and the interior-point optimizer has terminated due to maximum number of iterations, then basis identification is performed if this parameter has the value *MSK_ON*.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Interior-point method*, *Basis identification*

MSK_IPAR_BI_IGNORE_NUM_ERROR

If the parameter *MSK_IPAR_INTPNT_BASIS* has the value *MSK_BI_NO_ERROR* and the interior-point optimizer has terminated due to a numerical problem, then basis identification is performed if this parameter has the value *MSK_ON*.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Interior-point method*, *Basis identification*

MSK_IPAR_BI_MAX_ITERATIONS

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

Accepted Values: [0 ;+inf]

Default Value: 1000000

Groups: *Basis identification*, *Termination criterion*

MSK_IPAR_CACHE_LICENSE

Specifies if the license is kept checked out for the lifetime of the mosek environment (*MSK_ON*) or returned to the server immediately after the optimization (*MSK_OFF*).

By default the license is checked out for the lifetime of the **MOSEK** environment by the first call to the optimizer.

Check-in and check-out of licenses have an overhead. Frequent communication with the license server should be avoided.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *License manager*

MSK_IPAR_CHECK_CONVEXITY
Specify the level of convexity check on quadratic problems

Accepted Values: *MSKcheckconvexitytypee*

Default Value: *MSK_CHECK_CONVEXITY_FULL*

Groups: *Data check*, *Nonlinear convex method*

MSK_IPAR_COMPRESS_STATFILE
Control compression of stat files.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Infeasibility report*

MSK_IPAR_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.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Overall solver*

MSK_IPAR_INFEAS_REPORT_AUTO
Controls whether an infeasibility report is automatically produced after the optimization if the problem is primal or dual infeasible.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

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

MSK_IPAR_INFEAS_REPORT_LEVEL
Controls the amount of information presented in an infeasibility report. Higher values imply more information.

Accepted Values: [0 ;+inf]

Default Value: 1

Groups: *Infeasibility report*, *Output information*

MSK_IPAR_INTPNT_BASIS
Controls whether the interior-point optimizer also computes an optimal basis.

Accepted Values: *MSKbasindtypee*

Default Value: *MSK_BI_ALWAYS*

Groups: *Interior-point method*, *Basis identification*

MSK_IPAR_INTPNT_DIFF_STEP
Controls whether different step sizes are allowed in the primal and dual space.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Interior-point method*

MSK_IPAR_INTPNT_HOTSTART
Currently not in use.

Accepted Values: *MSKintpnthotstarte*

Default Value: *MSK_INTPNT_HOTSTART_NONE*

Groups: *Interior-point method*

MSK_IPAR_INTPNT_MAX_ITERATIONS
Controls the maximum number of iterations allowed in the interior-point optimizer.

Accepted Values: [0 ;+inf]

Default Value: 400

Groups: *Interior-point method*, *Termination criterion*

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.

Accepted Values: [-1 ;+inf]

Default Value: -1

Groups: *Interior-point method*

MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS
Maximum number of steps to be used by the iterative refinement of the search direction. A negative value implies that the optimizer chooses the maximum number of iterative refinement steps.

Accepted Values: [-inf ;+inf]

Default Value: -1

Groups: *Interior-point method*

MSK_IPAR_INTPNT_MULTI_THREAD
Controls whether the interior-point optimizers are allowed to employ multiple threads if more threads is available.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Optimization system*

MSK_IPAR_INTPNT_OFF_COL_TRH
Controls how many offending columns are detected in the Jacobian of the constraint matrix.

0	no detection
1	aggressive detection
> 1	higher values mean less aggressive detection

Accepted Values: [0 ;+inf]

Default Value: 40

Groups: *Interior-point method*

MSK_IPAR_INTPNT_ORDER_METHOD
Controls the ordering strategy used by the interior-point optimizer when factorizing the Newton equation system.

Accepted Values: *MSKorderingtypee*

Default Value: *MSK_ORDER_METHOD_FREE*

Groups: *Interior-point method*

MSK_IPAR_INTPNT_REGULARIZATION_USE
Controls whether regularization is allowed.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Interior-point method*

MSK_IPAR_INTPNT_SCALING
Controls how the problem is scaled before the interior-point optimizer is used.

Accepted Values: *MSKscalingtypee*

Default Value: *MSK_SCALING_FREE*

Groups: *Interior-point method*

MSK_IPAR_INTPNT_SOLVE_FORM
Controls whether the primal or the dual problem is solved.

Accepted Values: *MSKsolveforme*

Default Value: *MSK_SOLVE_FREE*

Groups: *Interior-point method*

MSK_IPAR_INTPNT_STARTING_POINT
Starting point used by the interior-point optimizer.

Accepted Values: *MSKstartpointtypee*

Default Value: *MSK_STARTING_POINT_FREE*

Groups: *Interior-point method*

MSK_IPAR_LICENSE_DEBUG
This option is used to turn on debugging of the license manager.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *License manager*

MSK_IPAR_LICENSE_PAUSE_TIME
If *MSK_IPAR_LICENSE_WAIT* = *MSK_ON* and no license is available, then **MOSEK** sleeps a number of milliseconds between each check of whether a license has become free.

Accepted Values: [0 ;1000000]

Default Value: 100

Groups: *License manager*

MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS
Controls whether license features expire warnings are suppressed.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *License manager*, *Output information*

MSK_IPAR_LICENSE_TRH_EXPIRY_WRN
If a license feature expires in a numbers days less than the value of this parameter then a warning will be issued.

Accepted Values: [0 ;+inf]

Default Value: 7

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.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Overall solver*, *Optimization system*, *License manager*

MSK_IPAR_LOG
Controls the amount of log information. The value 0 implies that all log information is suppressed. A higher level implies that more information is logged.

Please note that if a task is employed to solve a sequence of optimization problems the value of this parameter is reduced by the value of *MSK_IPAR_LOG_CUT_SECOND_OPT* for the second and any subsequent optimizations.

Accepted Values: [0 ;+inf]

Default Value: 10

Groups: *Output information*, *Logging*

MSK_IPAR_LOG_ANA_PRO
Controls amount of output from the problem analyzer.

Accepted Values: [0 ;+inf]

Default Value: 1

Groups: *Analysis*, *Logging*

MSK_IPAR_LOG_BI
Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.

Accepted Values: [0 ;+inf]

Default Value: 4

Groups: *Basis identification*, *Output information*, *Logging*

MSK_IPAR_LOG_BI_FREQ
Controls how frequent the optimizer outputs information about the basis identification and how frequent the user-defined call-back function is called.

Accepted Values: [0 ;+inf]

Default Value: 2500

Groups: *Basis identification*, *Output information*, *Logging*

MSK_IPAR_LOG_CHECK_CONVEXITY
Controls logging in convexity check on quadratic problems. Set to a positive value to turn logging on. If a quadratic coefficient matrix is found to violate the requirement of PSD (NSD) then a list of negative (positive) pivot elements is printed. The absolute value of the pivot elements is also shown.

Accepted Values: [0 ;+inf]

Default Value: 0

Groups: *Data check*, *Nonlinear convex method*

MSK_IPAR_LOG_CUT_SECOND_OPT
If a task is employed to solve a sequence of optimization problems, then the value of the log levels is reduced by the value of this parameter. E.g *MSK_IPAR_LOG* and *MSK_IPAR_LOG_SIM* are reduced by the value of this parameter for the second and any subsequent optimizations.

Accepted Values: [0 ;+inf]

Default Value: 1

Groups: *Output information*, *Logging*

MSK_IPAR_LOG_EXPAND
Controls the amount of logging when a data item such as the maximum number constrains is expanded.

Accepted Values: [0 ;+inf]

Default Value: 0

Groups: *Output information*, *Logging*

MSK_IPAR_LOG_FACTOR
If turned on, then the factor log lines are added to the log.

Accepted Values: [0 ;+inf]

Default Value: 1

Groups: *Output information*, *Logging*

MSK_IPAR_LOG_FEAS_REPAIR
Controls the amount of output printed when performing feasibility repair. A value higher than one means extensive logging.

Accepted Values: [0 ;+inf]

Default Value: 1

Groups: *Output information*, *Logging*

MSK_IPAR_LOG_FILE
If turned on, then some log info is printed when a file is written or read.

Accepted Values: [0 ;+inf]

Default Value: 1

Groups: *Data input/output*, *Output information*, *Logging*

MSK_IPAR_LOG_HEAD
If turned on, then a header line is added to the log.

Accepted Values: [0 ;+inf]

Default Value: 1

Groups: *Output information*, *Logging*

MSK_IPAR_LOG_INFEAS_ANA
Controls amount of output printed by the infeasibility analyzer procedures. A higher level implies that more information is logged.

Accepted Values: [0 ;+inf]

Default Value: 1

Groups: *Infeasibility report*, *Output information*, *Logging*

MSK_IPAR_LOG_INTPNT
Controls amount of output printed by the interior-point optimizer. A higher level implies that more information is logged.

Accepted Values: [0 ;+inf]

Default Value: 4

Groups: *Interior-point method*, *Output information*, *Logging*

MSK_IPAR_LOG_MIO
: Controls the log level for the mixed-integer optimizer. A higher level implies that more information is logged.

  Accepted Values: [0 ;+inf]

  Default Value: 4

  Groups: *Mixed-integer optimization*, *Output information*, *Logging*

MSK_IPAR_LOG_MIO_FREQ
: Controls how frequent the mixed-integer optimizer prints the log line. It will print line every time *MSK_IPAR_LOG_MIO_FREQ* relaxations have been solved.

  Accepted Values: [-inf ;+inf]

  Default Value: 10

  Groups: *Mixed-integer optimization*, *Output information*, *Logging*

MSK_IPAR_LOG_OPTIMIZER
: Controls the amount of general optimizer information that is logged.

  Accepted Values: [0 ;+inf]

  Default Value: 1

  Groups: *Output information*, *Logging*

MSK_IPAR_LOG_ORDER
: If turned on, then factor lines are added to the log.

  Accepted Values: [0 ;+inf]

  Default Value: 1

  Groups: *Output information*, *Logging*

MSK_IPAR_LOG_PRESOLVE
: Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.

  Accepted Values: [0 ;+inf]

  Default Value: 1

  Groups: *Interior-point method*, *Logging*

MSK_IPAR_LOG_RESPONSE
: Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.

  Accepted Values: [0 ;+inf]

  Default Value: 0

  Groups: *Output information*, *Logging*

MSK_IPAR_LOG_SENSITIVITY
: Controls the amount of logging during the sensitivity analysis.

  0.Means no logging information is produced.

  1.Timing information is printed.

  2.Sensitivity results are printed.

  Accepted Values: [0 ;+inf]

  Default Value: 1

  Groups: *Output information*, *Logging*

MSK_IPAR_LOG_SENSITIVITY_OPT
Controls the amount of logging from the optimizers employed during the sensitivity analysis. 0 means no logging information is produced.

Accepted Values: [0 ;+inf]

Default Value: 0

Groups: *Output information*, *Logging*

MSK_IPAR_LOG_SIM
Controls amount of output printed by the simplex optimizer. A higher level implies that more information is logged.

Accepted Values: [0 ;+inf]

Default Value: 4

Groups: *Simplex optimizer*, *Output information*, *Logging*

MSK_IPAR_LOG_SIM_FREQ
Controls how frequent the simplex optimizer outputs information about the optimization and how frequent the user-defined call-back function is called.

Accepted Values: [0 ;+inf]

Default Value: 1000

Groups: *Simplex optimizer*, *Output information*, *Logging*

MSK_IPAR_LOG_SIM_MINOR
Currently not in use.

Accepted Values: [0 ;+inf]

Default Value: 1

Groups: *Simplex optimizer*, *Output information*

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

Accepted Values: [0 ;+inf]

Default Value: 0

Groups: *Output information*, *Optimization system*, *Logging*

MSK_IPAR_MAX_NUM_WARNINGS
Each warning is shown a limit number times controlled by this parameter. A negative value is identical to infinite number of times.

Accepted Values: [-inf ;+inf]

Default Value: 10

Groups: *Output information*

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

Accepted Values: *MSKbranchdire*

Default Value: *MSK_BRANCH_DIR_FREE*

Groups: *Mixed-integer optimization*

MSK_IPAR_MIO_CONSTRUCT_SOL
If set to *MSK_ON* and all integer variables have been given a value for which a feasible mixed integer solution exists, then **MOSEK** generates an initial solution to the mixed integer problem by fixing all integer values and solving the remaining problem.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_CLIQUE

Controls whether clique cuts should be generated.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_CMIR

Controls whether mixed integer rounding cuts should be generated.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_GMI

Controls whether GMI cuts should be generated.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_IMPLIED_BOUND

Controls whether implied bound cuts should be generated.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_KNAPSACK_COVER

Controls whether knapsack cover cuts should be generated.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_SELECTION_LEVEL

Controls how aggressively generated cuts are selected to be included in the relaxation.

-1. The optimizer chooses the level of cut selection

0.Generated cuts less likely to be added to the relaxation

1.Cuts are more aggressively selected to be included in the relaxation

Accepted Values: [-1 ;+1]

Default Value: -1

Groups: *Mixed-integer optimization*

MSK_IPAR_MIO_HEURISTIC_LEVEL

Controls the heuristic employed by the mixed-integer optimizer to locate an initial good integer feasible solution. A value of zero means the heuristic is not used at all. A larger value than 0 means that a gradually more sophisticated heuristic is used which is computationally more expensive. A negative value implies that the optimizer chooses the heuristic. Normally a value around 3 to 5 should be optimal.

Accepted Values: [-inf ;+inf]

Default Value: -1

Groups: *Mixed-integer optimization*

MSK_IPAR_MIO_MAX_NUM_BRANCHES

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

Accepted Values: [-inf ;+inf]

Default Value: -1

Groups: *Mixed-integer optimization*, *Termination criterion*

MSK_IPAR_MIO_MAX_NUM_RELAXS

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

Accepted Values: [-inf ;+inf]

Default Value: -1

Groups: *Mixed-integer optimization*

MSK_IPAR_MIO_MAX_NUM_SOLUTIONS

The mixed-integer optimizer can be terminated after a certain number of different feasible solutions has been located. If this parameter has the value $n > 0$, then the mixed-integer optimizer will be terminated when n feasible solutions have been located.

Accepted Values: [-inf ;+inf]

Default Value: -1

Groups: *Mixed-integer optimization*, *Termination criterion*

MSK_IPAR_MIO_MODE

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

Accepted Values: *MSKmiomodee*

Default Value: *MSK_MIO_MODE_SATISFIED*

Groups: *Overall solver*

MSK_IPAR_MIO_MT_USER_CB

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Optimization system*

MSK_IPAR_MIO_NODE_OPTIMIZER

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

Accepted Values: *MSKoptimizertypee*

Default Value: *MSK_OPTIMIZER_FREE*

Groups: *Mixed-integer optimization*

MSK_IPAR_MIO_NODE_SELECTION

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

Accepted Values: *MSKmionodeseltypee*

Default Value: *MSK_MIO_NODE_SELECTION_FREE*

Groups: *Mixed-integer optimization*

`MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE`
Enables or disables perspective reformulation in presolve.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_ON*

`Groups:` *Mixed-integer optimization*

`MSK_IPAR_MIO_PROBING_LEVEL`
Controls the amount of probing employed by the mixed-integer optimizer in presolve.

-1. The optimizer chooses the level of probing employed

0.Probing is disabled

1.A low amount of probing is employed

2.A medium amount of probing is employed

3.A high amount of probing is employed

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

`Default Value:` -1

`Groups:` *Mixed-integer optimization*

`MSK_IPAR_MIO_RINS_MAX_NODES`
Controls the maximum number of nodes allowed in each call to the RINS heuristic. The default value of -1 means that the value is determined automatically. A value of zero turns off the heuristic.

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

`Default Value:` -1

`Groups:` *Mixed-integer optimization*

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

`Accepted Values:` *MSKoptimizertypee*

`Default Value:` *MSK_OPTIMIZER_FREE*

`Groups:` *Mixed-integer optimization*

`MSK_IPAR_MIO_ROOT_REPEAT_PRESOLVE_LEVEL`
Controls whether presolve can be repeated at root node.

- -1 The optimizer chooses whether presolve is repeated
- 0 Never repeat presolve
- 1 Always repeat presolve

`Accepted Values:` [-1 ;1]

`Default Value:` -1

`Groups:` *Mixed-integer optimization*

`MSK_IPAR_MIO_VB_DETECTION_LEVEL`
Controls how much effort is put into detecting variable bounds.

-1. The optimizer chooses

0.No variable bounds are detected

1.Only detect variable bounds that are directly represented in the problem

2.Detect variable bounds in probing

Accepted Values: [-1 ;+2]

Default Value: -1

Groups: *Mixed-integer optimization*

MSK_IPAR_MT_SPINCOUNT
Set the number of iterations to spin before sleeping.

Accepted Values: [0 ;1000000000]

Default Value: 0

Groups: *Optimization system*

MSK_IPAR_NUM_THREADS
Controls the number of threads employed by the optimizer. If set to 0 the number of threads used will be equal to the number of cores detected on the machine.

Accepted Values: [0 ;+inf]

Default Value: 0

Groups: *Optimization system*

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

Accepted Values: [0 ;+inf]

Default Value: 5

Groups: *Data input/output*

MSK_IPAR_OPF_WRITE_HEADER
Write a text header with date and **MOSEK** version in an OPF file.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Data input/output*

MSK_IPAR_OPF_WRITE_HINTS
Write a hint section with problem dimensions in the beginning of an OPF file.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Data input/output*

MSK_IPAR_OPF_WRITE_PARAMETERS
Write a parameter section in an OPF file.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Data input/output*

MSK_IPAR_OPF_WRITE_PROBLEM
Write objective, constraints, bounds etc. to an OPF file.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Data input/output*

MSK_IPAR_OPF_WRITE_SOLUTIONS
Enable inclusion of solutions in the OPF files.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Data input/output*

MSK_IPAR_OPF_WRITE_SOL_BAS

If *MSK_IPAR_OPF_WRITE_SOLUTIONS* is *MSK_ON* and a basic solution is defined, include the basic solution in OPF files.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Data input/output*

MSK_IPAR_OPF_WRITE_SOL_ITG

If *MSK_IPAR_OPF_WRITE_SOLUTIONS* is *MSK_ON* and an integer solution is defined, write the integer solution in OPF files.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Data input/output*

MSK_IPAR_OPF_WRITE_SOL_ITR

If *MSK_IPAR_OPF_WRITE_SOLUTIONS* is *MSK_ON* and an interior solution is defined, write the interior solution in OPF files.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Data input/output*

MSK_IPAR_OPTIMIZER

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

Accepted Values: *MSKoptimizertypee*

Default Value: *MSK_OPTIMIZER_FREE*

Groups: *Overall solver*

MSK_IPAR_PARAM_READ_CASE_NAME

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Data input/output*

MSK_IPAR_PARAM_READ_IGN_ERROR

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Data input/output*

MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_FILL

Controls the maximum amount of fill-in that can be created by one pivot in the elimination phase of the presolve. A negative value menas the parameter value is selected automatically.

Accepted Values: [-inf ;+inf]

Default Value: -1

Groups: *Presolve*

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

Accepted Values: [-inf ;+inf]

Default Value: -1

Groups: *Presolve*

MSK_IPAR_PRESOLVE_LEVEL
Currently not used.

Accepted Values: [-inf ;+inf]

Default Value: -1

Groups: *Overall solver*, *Presolve*

MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH
The linear dependency check is potentially computationally expensive.

Accepted Values: [-inf ;+inf]

Default Value: 100

Groups: *Presolve*

MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH
The linear dependency check is potentially computationally expensive.

Accepted Values: [-inf ;+inf]

Default Value: 100

Groups: *Presolve*

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Presolve*

MSK_IPAR_PRESOLVE_MAX_NUM_REDUCTIONS
Controls the maximum number of reductions performed by the presolve. The value of the parameter is normally only changed in connection with debugging. A negative value implies that an infinite number of reductions are allowed.

Accepted Values: [-inf ;+inf]

Default Value: -1

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

Accepted Values: *MSKpresolvemodee*

Default Value: *MSK_PRESOLVE_MODE_FREE*

Groups: *Overall solver*, *Presolve*

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

Accepted Values: *MSKoptimizertypee*

Default Value: *MSK_OPTIMIZER_FREE*

Groups: *Overall solver*

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

Accepted Values: *MSKcompresstypee*

Default Value: *MSK_COMPRESS_FREE*

Groups: *Data input/output*

MSK_IPAR_READ_DATA_FORMAT
Format of the data file to be read.

Accepted Values: *MSKdataformate*

Default Value: *MSK_DATA_FORMAT_EXTENSION*

Groups: *Data input/output*

MSK_IPAR_READ_DEBUG
Turns on additional debugging information when reading files.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Data input/output*

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Data input/output*

MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU
If this option is turned on, **MOSEK** will drop variables that are defined for the first time in the bounds section.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Data input/output*

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Data input/output*

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

Accepted Values: *MSKmpsformate*

Default Value: *MSK_MPS_FORMAT_FREE*

Groups: *Data input/output*

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

Accepted Values: [80 ;+inf]

Default Value: 1024

Groups: *Data input/output*

MSK_IPAR_READ_TASK_IGNORE_PARAM
Controls whether **MOSEK** should ignore the parameter setting defined in the task file and use the default parameter setting instead.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Data input/output*

MSK_IPAR_SENSITIVITY_ALL
Not applicable.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Overall solver*

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

Accepted Values: *MSKoptimizertypee*

Default Value: *MSK_OPTIMIZER_FREE_SIMPLEX*

Groups: *Overall solver*, *Simplex optimizer*

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

Accepted Values: *MSKsensitivitytypee*

Default Value: *MSK_SENSITIVITY_TYPE_BASIS*

Groups: *Overall solver*

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Simplex optimizer*

MSK_IPAR_SIM_DEGEN
Controls how aggressively degeneration is handled.

Accepted Values: *MSKsimdegene*

Default Value: *MSK_SIM_DEGEN_FREE*

Groups: *Simplex optimizer*

MSK_IPAR_SIM_DUAL_CRASH
Controls whether crashing is performed in the dual simplex optimizer.

If this parameter is set to x, then a crash will be performed if a basis consists of more than $(100-x)$ mod f_v entries, where f_v is the number of fixed variables.

Accepted Values: [0 ;+inf]

Default Value: 90

Groups: *Dual simplex optimizer*

MSK_IPAR_SIM_DUAL_PHASEONE_METHOD
An experimental feature.

Accepted Values: [0 ;10]

Default Value: 0

Groups: *Simplex optimizer*

MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION
The dual simplex optimizer can use a so-called restricted selection/pricing strategy to chooses the outgoing variable. Hence, if restricted selection is applied, then the dual simplex optimizer first choose a subset of all the potential outgoing variables. Next, for some time it will choose the outgoing variable only among the subset. From time to time the subset is redefined.

A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.

Accepted Values: [0 ;100]

Default Value: 50

Groups: *Dual simplex optimizer*

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

Accepted Values: *MSKsimseltypee*

Default Value: *MSK_SIM_SELECTION_FREE*

Groups: *Dual simplex optimizer*

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

Accepted Values: *MSKsimdupvece*

Default Value: *MSK_SIM_EXPLOIT_DUPVEC_OFF*

Groups: *Simplex optimizer*

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

Accepted Values: *MSKsimhotstarte*

Default Value: *MSK_SIM_HOTSTART_FREE*

Groups: *Simplex optimizer*

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

MSK_IPAR_SIM_INTEGER
An experimental feature.

Accepted Values: [0 ;10]

Default Value: 0

Groups: *Simplex optimizer*

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

Accepted Values: [0 ;+inf]

Default Value: 10000000

Groups: *Simplex optimizer*, *Termination criterion*

MSK_IPAR_SIM_MAX_NUM_SETBACKS

Controls how many set-backs are allowed within a simplex optimizer. A set-back is an event where the optimizer moves in the wrong direction. This is impossible in theory but may happen due to numerical problems.

Accepted Values: [0 ;+inf]

Default Value: 250

Groups: *Simplex optimizer*

MSK_IPAR_SIM_NON_SINGULAR

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Simplex optimizer*

MSK_IPAR_SIM_PRIMAL_CRASH

Controls whether crashing is performed in the primal simplex optimizer.

In general, if a basis consists of more than (100-this parameter value)% fixed variables, then a crash will be performed.

Accepted Values: [0 ;+inf]

Default Value: 90

Groups: *Primal simplex optimizer*

MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD

An experimental feature.

Accepted Values: [0 ;10]

Default Value: 0

Groups: *Simplex optimizer*

MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION

The primal simplex optimizer can use a so-called restricted selection/pricing strategy to chooses the outgoing variable. Hence, if restricted selection is applied, then the primal simplex optimizer first choose a subset of all the potential incoming variables. Next, for some time it will choose the incoming variable only among the subset. From time to time the subset is redefined.

A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.

Accepted Values: [0 ;100]

Default Value: 50

Groups: *Primal simplex optimizer*

MSK_IPAR_SIM_PRIMAL_SELECTION

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

Accepted Values: *MSKsimseltypee*

Default Value: *MSK_SIM_SELECTION_FREE*

Groups: *Primal simplex optimizer*

MSK_IPAR_SIM_REFACTOR_FREQ

Controls how frequent the basis is refactorized. The value 0 means that the optimizer determines the best point of refactorization.

It is strongly recommended NOT to change this parameter.

Accepted Values: [0 ;+inf]

Default Value: 0

Groups: *Simplex optimizer*

MSK_IPAR_SIM_REFORMULATION

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

Accepted Values: *MSKsimreforme*

Default Value: *MSK_SIM_REFORMULATION_OFF*

Groups: *Simplex optimizer*

MSK_IPAR_SIM_SAVE_LU

Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Simplex optimizer*

MSK_IPAR_SIM_SCALING

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

Accepted Values: *MSKscalingtypee*

Default Value: *MSK_SCALING_FREE*

Groups: *Simplex optimizer*

MSK_IPAR_SIM_SCALING_METHOD

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

Accepted Values: *MSKscalingmethode*

Default Value: *MSK_SCALING_METHOD_POW2*

Groups: *Simplex optimizer*

MSK_IPAR_SIM_SOLVE_FORM

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

Accepted Values: *MSKsolveforme*

Default Value: *MSK_SOLVE_FREE*

Groups: *Simplex optimizer*

MSK_IPAR_SIM_STABILITY_PRIORITY

Controls how high priority the numerical stability should be given.

Accepted Values: [0 ;100]

Default Value: 50

Groups: *Simplex optimizer*

MSK_IPAR_SIM_SWITCH_OPTIMIZER

The simplex optimizer sometimes chooses to solve the dual problem instead of the primal problem. This implies that if you have chosen to use the dual simplex optimizer and the problem is dualized, then it actually makes sense to use the primal simplex optimizer instead. If this parameter is on and the problem is dualized and furthermore the simplex optimizer is chosen to be the primal (dual) one, then it is switched to the dual (primal).

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Simplex optimizer*

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Progress call-back*, *Overall solver*

MSK_IPAR_SOL_FILTER_KEEP_BASIC
If turned on, then basic and super basic constraints and variables are written to the solution file independent of the filter setting.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Solution input/output*

MSK_IPAR_SOL_FILTER_KEEP_RANGED
If turned on, then ranged constraints and variables are written to the solution file independent of the filter setting.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Solution input/output*

MSK_IPAR_SOL_READ_NAME_WIDTH
When a solution is read by **MOSEK** and some constraint, variable or cone names contain blanks, then a maximum name width much be specified. A negative value implies that no name contain blanks.

Accepted Values: [-inf ;+inf]

Default Value: -1

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

MSK_IPAR_SOL_READ_WIDTH
Controls the maximal acceptable width of line in the solutions when read by **MOSEK**.

Accepted Values: [0 ;+inf]

Default Value: 1024

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

MSK_IPAR_TIMING_LEVEL
Controls the a amount of timing performed inside **MOSEK**.

Accepted Values: [0 ;+inf]

Default Value: 1

Groups: *Optimization system*

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

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

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

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

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

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

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

Accepted Values: [0 ;+inf]

Default Value: 0

Groups: *Data input/output*

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

Accepted Values: *MSKdataformate*

Default Value: *MSK_DATA_FORMAT_EXTENSION*

Groups: *Data input/output*

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Data input/output*

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Data input/output*

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Data input/output*

MSK_IPAR_WRITE_GENERIC_NAMES_IO
Index origin used in generic names.

Accepted Values: [0 ;+inf]

Default Value: 1

Groups: *Data input/output*

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Data input/output*

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

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

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

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

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

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

MSK_IPAR_WRITE_LP_FULL_OBJ
Write all variables, including the ones with 0-coefficients, in the objective.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Data input/output*

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

Accepted Values: [40 ;+inf]

Default Value: 80

Groups: *Data input/output*

MSK_IPAR_WRITE_LP_QUOTED_NAMES
If this option is turned on, then **MOSEK** will quote invalid LP names when writing an LP file.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Data input/output*

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

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_OFF*

Groups: *Data input/output*

MSK_IPAR_WRITE_LP_TERMS_PER_LINE
Maximum number of terms on a single line in an LP file written by **MOSEK**. 0 means unlimited.

Accepted Values: [0 ;+inf]

Default Value: 10

Groups: *Data input/output*

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

`Accepted Values:` *MSKmpsformate*

`Default Value:` *MSK_MPS_FORMAT_FREE*

`Groups:` *Data input/output*

`MSK_IPAR_WRITE_MPS_INT`
Controls if marker records are written to the MPS file to indicate whether variables are integer restricted.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_ON*

`Groups:` *Data input/output*

`MSK_IPAR_WRITE_PRECISION`
Controls the precision with which `double` numbers are printed in the MPS data file. In general it is not worthwhile to use a value higher than 15.

`Accepted Values:` [0 ;+inf]

`Default Value:` 15

`Groups:` *Data input/output*

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

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_ON*

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

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

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_ON*

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

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

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_ON*

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

`MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES`
Even if the names are invalid MPS names, then they are employed when writing the solution file.

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_OFF*

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

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

`Accepted Values:` *MSKonoffkeye*

`Default Value:` *MSK_ON*

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

MSK_IPAR_WRITE_TASK_INC_SOL
Controls whether the solutions are stored in the task file too.

Accepted Values: *MSKonoffkeye*

Default Value: *MSK_ON*

Groups: *Data input/output*

MSK_IPAR_WRITE_XML_MODE
Controls if linear coefficients should be written by row or column when writing in the XML file format.

Accepted Values: *MSKxmlwriteroutputtypee*

Default Value: *MSK_WRITE_XML_MODE_ROW*

Groups: *Data input/output*

String Parameters

MSK_SPAR_BAS_SOL_FILE_NAME
Name of the `bas` solution file.

Accepted Values: Any valid file name.

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

MSK_SPAR_DATA_FILE_NAME
Data are read and written to this file.

Accepted Values: Any valid file name.

Groups: *Data input/output*

MSK_SPAR_DEBUG_FILE_NAME
MOSEK debug file.

Accepted Values: Any valid file name.

Groups: *Data input/output*

MSK_SPAR_INT_SOL_FILE_NAME
Name of the `int` solution file.

Accepted Values: Any valid file name.

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

MSK_SPAR_ITR_SOL_FILE_NAME
Name of the `itr` solution file.

Accepted Values: Any valid file name.

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

MSK_SPAR_MIO_DEBUG_STRING
For internal use only.

Accepted Values: Any valid string.

Groups: *Data input/output*

MSK_SPAR_PARAM_COMMENT_SIGN
Only the first character in this string is used. It is considered as a start of comment sign in the **MOSEK** parameter file. Spaces are ignored in the string.

Accepted Values: Any valid string.

Default Value: %%

Groups: *Data input/output*

MSK_SPAR_PARAM_READ_FILE_NAME

Modifications to the parameter database is read from this file.

Accepted Values: Any valid file name.

Groups: *Data input/output*

MSK_SPAR_PARAM_WRITE_FILE_NAME

The parameter database is written to this file.

Accepted Values: Any valid file name.

Groups: *Data input/output*

MSK_SPAR_READ_MPS_BOU_NAME

Name of the BOUNDS vector used. An empty name means that the first BOUNDS vector is used.

Accepted Values: Any valid MPS name.

Groups: *Data input/output*

MSK_SPAR_READ_MPS_OBJ_NAME

Name of the free constraint used as objective function. An empty name means that the first constraint is used as objective function.

Accepted Values: Any valid MPS name.

Groups: *Data input/output*

MSK_SPAR_READ_MPS_RAN_NAME

Name of the RANGE vector used. An empty name means that the first RANGE vector is used.

Accepted Values: Any valid MPS name.

Groups: *Data input/output*

MSK_SPAR_READ_MPS_RHS_NAME

Name of the RHS used. An empty name means that the first RHS vector is used.

Accepted Values: Any valid MPS name.

Groups: *Data input/output*

MSK_SPAR_REMOTE_ACCESS_TOKEN

An access token used to submit tasks to a remote **MOSEK** server. An access token is a random 32-byte string encoded in base64, i.e. it is a 44 character ASCII string.

Accepted Values: Any valid string.

MSK_SPAR_SENSITIVITY_FILE_NAME

If defined, **MOSEK** reads this file as a sensitivity analysis data file specifying the type of analysis to be done.

Accepted Values: Any valid string.

Groups: *Data input/output*

MSK_SPAR_SENSITIVITY_RES_FILE_NAME

Accepted Values: Any valid string.

Groups: *Data input/output*

MSK_SPAR_SOL_FILTER_XC_LOW

A filter used to determine which constraints should be listed in the solution file. A value of 0.5 means that all constraints having `xc[i]>0.5` should be listed, whereas `+0.5` means that all constraints having `xc[i]>=blc[i]+0.5` should be listed. An empty filter means that no filter is applied.

Accepted Values: Any valid filter.

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

`MSK_SPAR_SOL_FILTER_XC_UPR`

A filter used to determine which constraints should be listed in the solution file. A value of `0.5` means that all constraints having `xc[i]<0.5` should be listed, whereas `-0.5` means all constraints having `xc[i]<=buc[i]-0.5` should be listed. An empty filter means that no filter is applied.

`Accepted Values:` Any valid filter.

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

`MSK_SPAR_SOL_FILTER_XX_LOW`

A filter used to determine which variables should be listed in the solution file. A value of "0.5" means that all constraints having `xx[j]>=0.5` should be listed, whereas "+0.5" means that all constraints having `xx[j]>=blx[j]+0.5` should be listed. An empty filter means no filter is applied.

`Accepted Values:` Any valid filter.

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

`MSK_SPAR_SOL_FILTER_XX_UPR`

A filter used to determine which variables should be listed in the solution file. A value of "0.5" means that all constraints having `xx[j]<0.5` should be printed, whereas "-0.5" means all constraints having `xx[j]<=bux[j]-0.5` should be listed. An empty filter means no filter is applied.

`Accepted Values:` Any valid file name.

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

`MSK_SPAR_STAT_FILE_NAME`

Statistics file name.

`Accepted Values:` Any valid file name.

`Groups:` *Data input/output*

`MSK_SPAR_STAT_KEY`

Key used when writing the summary file.

`Accepted Values:` Any valid XML string.

`Groups:` *Data input/output*

`MSK_SPAR_STAT_NAME`

Name used when writing the statistics file.

`Accepted Values:` Any valid XML string.

`Groups:` *Data input/output*

`MSK_SPAR_WRITE_LP_GEN_VAR_NAME`

Sometimes when an LP file is written additional variables must be inserted. They will have the prefix denoted by this parameter.

`Accepted Values:` Any valid string.

`Default Value:` xmskgen

`Groups:` *Data input/output*

10.2.2 Logging parameters.

- *MSK_IPAR_LOG*
- *MSK_IPAR_LOG_ANA_PRO*
- *MSK_IPAR_LOG_BI*
- *MSK_IPAR_LOG_BI_FREQ*
- *MSK_IPAR_LOG_CUT_SECOND_OPT*

- *MSK_IPAR_LOG_EXPAND*
- *MSK_IPAR_LOG_FACTOR*
- *MSK_IPAR_LOG_FEAS_REPAIR*
- *MSK_IPAR_LOG_FILE*
- *MSK_IPAR_LOG_HEAD*
- *MSK_IPAR_LOG_INFEAS_ANA*
- *MSK_IPAR_LOG_INTPNT*
- *MSK_IPAR_LOG_MIO*
- *MSK_IPAR_LOG_MIO_FREQ*
- *MSK_IPAR_LOG_OPTIMIZER*
- *MSK_IPAR_LOG_ORDER*
- *MSK_IPAR_LOG_PRESOLVE*
- *MSK_IPAR_LOG_RESPONSE*
- *MSK_IPAR_LOG_SENSITIVITY*
- *MSK_IPAR_LOG_SENSITIVITY_OPT*
- *MSK_IPAR_LOG_SIM*
- *MSK_IPAR_LOG_SIM_FREQ*
- *MSK_IPAR_LOG_STORAGE*

10.2.3 Basis identification parameters.

- *MSK_IPAR_BI_CLEAN_OPTIMIZER*
- *MSK_IPAR_BI_IGNORE_MAX_ITER*
- *MSK_IPAR_BI_IGNORE_NUM_ERROR*
- *MSK_IPAR_BI_MAX_ITERATIONS*
- *MSK_IPAR_INTPNT_BASIS*
- *MSK_IPAR_LOG_BI*
- *MSK_IPAR_LOG_BI_FREQ*
- *MSK_DPAR_SIM_LU_TOL_REL_PIV*

10.2.4 Output information parameters.

- *MSK_IPAR_INFEAS_REPORT_LEVEL*
- *MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS*
- *MSK_IPAR_LOG*
- *MSK_IPAR_LOG_BI*
- *MSK_IPAR_LOG_BI_FREQ*
- *MSK_IPAR_LOG_CUT_SECOND_OPT*
- *MSK_IPAR_LOG_EXPAND*
- *MSK_IPAR_LOG_FACTOR*

- *MSK_IPAR_LOG_FEAS_REPAIR*
- *MSK_IPAR_LOG_FILE*
- *MSK_IPAR_LOG_HEAD*
- *MSK_IPAR_LOG_INFEAS_ANA*
- *MSK_IPAR_LOG_INTPNT*
- *MSK_IPAR_LOG_MIO*
- *MSK_IPAR_LOG_MIO_FREQ*
- *MSK_IPAR_LOG_OPTIMIZER*
- *MSK_IPAR_LOG_ORDER*
- *MSK_IPAR_LOG_RESPONSE*
- *MSK_IPAR_LOG_SENSITIVITY*
- *MSK_IPAR_LOG_SENSITIVITY_OPT*
- *MSK_IPAR_LOG_SIM*
- *MSK_IPAR_LOG_SIM_FREQ*
- *MSK_IPAR_LOG_SIM_MINOR*
- *MSK_IPAR_LOG_STORAGE*
- *MSK_IPAR_MAX_NUM_WARNINGS*

10.2.5 Infeasibility report parameters.

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

10.2.6 Data check parameters.

- *MSK_IPAR_CHECK_CONVEXITY*
- *MSK_DPAR_DATA_SYM_MAT_TOL*
- *MSK_DPAR_DATA_SYM_MAT_TOL_HUGE*
- *MSK_DPAR_DATA_SYM_MAT_TOL_LARGE*
- *MSK_DPAR_DATA_TOL_AIJ*
- *MSK_DPAR_DATA_TOL_AIJ_HUGE*
- *MSK_DPAR_DATA_TOL_AIJ_LARGE*
- *MSK_DPAR_DATA_TOL_BOUND_INF*
- *MSK_DPAR_DATA_TOL_BOUND_WRN*
- *MSK_DPAR_DATA_TOL_C_HUGE*
- *MSK_DPAR_DATA_TOL_CJ_LARGE*
- *MSK_DPAR_DATA_TOL_QIJ*
- *MSK_DPAR_DATA_TOL_X*
- *MSK_IPAR_LOG_CHECK_CONVEXITY*

- *MSK_DPAR_SEMIDEFINITE_TOL_APPROX*

10.2.7 Conic interior-point method parameters.

- *MSK_DPAR_INTPNT_CO_TOL_DFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_INFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_MU_RED*
- *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_CO_TOL_PFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_REL_GAP*

10.2.8 Primal simplex optimizer parameters.

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

10.2.9 Nonlinear convex method parameters.

- *MSK_IPAR_CHECK_CONVEXITY*
- *MSK_DPAR_INTPNT_NL_MERIT_BAL*
- *MSK_DPAR_INTPNT_NL_TOL_DFEAS*
- *MSK_DPAR_INTPNT_NL_TOL_MU_RED*
- *MSK_DPAR_INTPNT_NL_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_NL_TOL_PFEAS*
- *MSK_DPAR_INTPNT_NL_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_NL_TOL_REL_STEP*
- *MSK_DPAR_INTPNT_TOL_INFEAS*
- *MSK_IPAR_LOG_CHECK_CONVEXITY*

10.2.10 Optimization system parameters.

- *MSK_IPAR_AUTO_UPDATE_SOL_INFO*
- *MSK_IPAR_INTPNT_MULTI_THREAD*
- *MSK_IPAR_LICENSE_WAIT*
- *MSK_IPAR_LOG_STORAGE*
- *MSK_IPAR_MIO_MT_USER_CB*
- *MSK_IPAR_MT_SPINCOUNT*
- *MSK_IPAR_NUM_THREADS*
- *MSK_IPAR_TIMING_LEVEL*

10.2.11 Progress call-back parameters.

- *MSK_IPAR_SOLUTION_CALLBACK*

10.2.12 Overall solver parameters.

- *MSK_IPAR_BI_CLEAN_OPTIMIZER*
- *MSK_IPAR_INFEAS_PREFER_PRIMAL*
- *MSK_IPAR_LICENSE_WAIT*
- *MSK_IPAR_MIO_MODE*
- *MSK_IPAR_OPTIMIZER*
- *MSK_IPAR_PRESOLVE_LEVEL*
- *MSK_IPAR_PRESOLVE_USE*
- *MSK_IPAR_PRIMAL_REPAIR_OPTIMIZER*
- *MSK_IPAR_SENSITIVITY_ALL*
- *MSK_IPAR_SENSITIVITY_OPTIMIZER*
- *MSK_IPAR_SENSITIVITY_TYPE*
- *MSK_IPAR_SOLUTION_CALLBACK*

10.2.13 Simplex optimizer parameters.

- *MSK_DPAR_BASIS_REL_TOL_S*
- *MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE*
- *MSK_DPAR_BASIS_TOL_S*
- *MSK_DPAR_BASIS_TOL_X*
- *MSK_IPAR_LOG_SIM*
- *MSK_IPAR_LOG_SIM_FREQ*
- *MSK_IPAR_LOG_SIM_MINOR*
- *MSK_IPAR_SENSITIVITY_OPTIMIZER*
- *MSK_IPAR_SIM_BASIS_FACTOR_USE*
- *MSK_IPAR_SIM_DEGEN*
- *MSK_IPAR_SIM_DUAL_PHASEONE_METHOD*
- *MSK_IPAR_SIM_EXPLOIT_DUPVEC*
- *MSK_IPAR_SIM_HOTSTART*
- *MSK_IPAR_SIM_INTEGER*
- *MSK_DPAR_SIM_LU_TOL_REL_PIV*
- *MSK_IPAR_SIM_MAX_ITERATIONS*
- *MSK_IPAR_SIM_MAX_NUM_SETBACKS*
- *MSK_IPAR_SIM_NON_SINGULAR*
- *MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD*
- *MSK_IPAR_SIM_REFACTOR_FREQ*

- *MSK_IPAR_SIM_REFORMULATION*
- *MSK_IPAR_SIM_SAVE_LU*
- *MSK_IPAR_SIM_SCALING*
- *MSK_IPAR_SIM_SCALING_METHOD*
- *MSK_IPAR_SIM_SOLVE_FORM*
- *MSK_IPAR_SIM_STABILITY_PRIORITY*
- *MSK_IPAR_SIM_SWITCH_OPTIMIZER*
- *MSK_DPAR_SIMPLEX_ABS_TOL_PIV*

10.2.14 Solution input/output parameters.

- *MSK_SPAR_BAS_SOL_FILE_NAME*
- *MSK_IPAR_INFEAS_REPORT_AUTO*
- *MSK_SPAR_INT_SOL_FILE_NAME*
- *MSK_SPAR_ITR_SOL_FILE_NAME*
- *MSK_IPAR_SOL_FILTER_KEEP_BASIC*
- *MSK_IPAR_SOL_FILTER_KEEP_RANGED*
- *MSK_SPAR_SOL_FILTER_XC_LOW*
- *MSK_SPAR_SOL_FILTER_XC_UPR*
- *MSK_SPAR_SOL_FILTER_XX_LOW*
- *MSK_SPAR_SOL_FILTER_XX_UPR*
- *MSK_IPAR_SOL_READ_NAME_WIDTH*
- *MSK_IPAR_SOL_READ_WIDTH*
- *MSK_IPAR_WRITE_BAS_CONSTRAINTS*
- *MSK_IPAR_WRITE_BAS_HEAD*
- *MSK_IPAR_WRITE_BAS_VARIABLES*
- *MSK_IPAR_WRITE_INT_CONSTRAINTS*
- *MSK_IPAR_WRITE_INT_HEAD*
- *MSK_IPAR_WRITE_INT_VARIABLES*
- *MSK_IPAR_WRITE_SOL_BARVARIABLES*
- *MSK_IPAR_WRITE_SOL_CONSTRAINTS*
- *MSK_IPAR_WRITE_SOL_HEAD*
- *MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES*
- *MSK_IPAR_WRITE_SOL_VARIABLES*

10.2.15 Debugging parameters.

- *MSK_IPAR_AUTO_SORT_A_BEFORE_OPT*

10.2.16 Termination criterion parameters.

- *MSK_DPAR_BASIS_REL_TOL_S*
- *MSK_DPAR_BASIS_TOL_S*
- *MSK_DPAR_BASIS_TOL_X*
- *MSK_IPAR_BI_MAX_ITERATIONS*
- *MSK_DPAR_INTPNT_CO_TOL_DFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_INFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_MU_RED*
- *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_CO_TOL_PFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_REL_GAP*
- *MSK_IPAR_INTPNT_MAX_ITERATIONS*
- *MSK_DPAR_INTPNT_NL_TOL_DFEAS*
- *MSK_DPAR_INTPNT_NL_TOL_MU_RED*
- *MSK_DPAR_INTPNT_NL_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_NL_TOL_PFEAS*
- *MSK_DPAR_INTPNT_NL_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_QO_TOL_DFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_INFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_MU_RED*
- *MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_QO_TOL_PFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_TOL_DFEAS*
- *MSK_DPAR_INTPNT_TOL_INFEAS*
- *MSK_DPAR_INTPNT_TOL_MU_RED*
- *MSK_DPAR_INTPNT_TOL_PFEAS*
- *MSK_DPAR_INTPNT_TOL_REL_GAP*
- *MSK_DPAR_LOWER_OBJ_CUT*
- *MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH*
- *MSK_DPAR_MIO_DISABLE_TERM_TIME*
- *MSK_IPAR_MIO_MAX_NUM_BRANCHES*
- *MSK_IPAR_MIO_MAX_NUM_SOLUTIONS*
- *MSK_DPAR_MIO_MAX_TIME*
- *MSK_DPAR_MIO_NEAR_TOL_REL_GAP*
- *MSK_DPAR_MIO_REL_GAP_CONST*
- *MSK_DPAR_MIO_TOL_REL_GAP*
- *MSK_DPAR_OPTIMIZER_MAX_TIME*
- *MSK_IPAR_SIM_MAX_ITERATIONS*

- *MSK_DPAR_UPPER_OBJ_CUT*
- *MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH*

10.2.17 License manager parameters.

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

10.2.18 Dual simplex optimizer parameters.

- *MSK_IPAR_SIM_DUAL_CRASH*
- *MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION*
- *MSK_IPAR_SIM_DUAL_SELECTION*

10.2.19 Data input/output parameters.

- *MSK_SPAR_BAS_SOL_FILE_NAME*
- *MSK_SPAR_DATA_FILE_NAME*
- *MSK_SPAR_DEBUG_FILE_NAME*
- *MSK_IPAR_INFEAS_REPORT_AUTO*
- *MSK_SPAR_INT_SOL_FILE_NAME*
- *MSK_SPAR_ITR_SOL_FILE_NAME*
- *MSK_IPAR_LOG_FILE*
- *MSK_SPAR_MIO_DEBUG_STRING*
- *MSK_IPAR_OPF_MAX_TERMS_PER_LINE*
- *MSK_IPAR_OPF_WRITE_HEADER*
- *MSK_IPAR_OPF_WRITE_HINTS*
- *MSK_IPAR_OPF_WRITE_PARAMETERS*
- *MSK_IPAR_OPF_WRITE_PROBLEM*
- *MSK_IPAR_OPF_WRITE_SOL_BAS*
- *MSK_IPAR_OPF_WRITE_SOL_ITG*
- *MSK_IPAR_OPF_WRITE_SOL_ITR*
- *MSK_IPAR_OPF_WRITE_SOLUTIONS*
- *MSK_SPAR_PARAM_COMMENT_SIGN*
- *MSK_IPAR_PARAM_READ_CASE_NAME*
- *MSK_SPAR_PARAM_READ_FILE_NAME*
- *MSK_IPAR_PARAM_READ_IGN_ERROR*
- *MSK_SPAR_PARAM_WRITE_FILE_NAME*

- *MSK_IPAR_READ_DATA_COMPRESSED*
- *MSK_IPAR_READ_DATA_FORMAT*
- *MSK_IPAR_READ_DEBUG*
- *MSK_IPAR_READ_KEEP_FREE_CON*
- *MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU*
- *MSK_IPAR_READ_LP_QUOTED_NAMES*
- *MSK_SPAR_READ_MPS_BOU_NAME*
- *MSK_IPAR_READ_MPS_FORMAT*
- *MSK_SPAR_READ_MPS_OBJ_NAME*
- *MSK_SPAR_READ_MPS_RAN_NAME*
- *MSK_SPAR_READ_MPS_RHS_NAME*
- *MSK_IPAR_READ_MPS_WIDTH*
- *MSK_IPAR_READ_TASK_IGNORE_PARAM*
- *MSK_SPAR_SENSITIVITY_FILE_NAME*
- *MSK_SPAR_SENSITIVITY_RES_FILE_NAME*
- *MSK_SPAR_SOL_FILTER_XC_LOW*
- *MSK_SPAR_SOL_FILTER_XC_UPR*
- *MSK_SPAR_SOL_FILTER_XX_LOW*
- *MSK_SPAR_SOL_FILTER_XX_UPR*
- *MSK_IPAR_SOL_READ_NAME_WIDTH*
- *MSK_IPAR_SOL_READ_WIDTH*
- *MSK_SPAR_STAT_FILE_NAME*
- *MSK_SPAR_STAT_KEY*
- *MSK_SPAR_STAT_NAME*
- *MSK_IPAR_WRITE_BAS_CONSTRAINTS*
- *MSK_IPAR_WRITE_BAS_HEAD*
- *MSK_IPAR_WRITE_BAS_VARIABLES*
- *MSK_IPAR_WRITE_DATA_COMPRESSED*
- *MSK_IPAR_WRITE_DATA_FORMAT*
- *MSK_IPAR_WRITE_DATA_PARAM*
- *MSK_IPAR_WRITE_FREE_CON*
- *MSK_IPAR_WRITE_GENERIC_NAMES*
- *MSK_IPAR_WRITE_GENERIC_NAMES_IO*
- *MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_ITEMS*
- *MSK_IPAR_WRITE_INT_CONSTRAINTS*
- *MSK_IPAR_WRITE_INT_HEAD*
- *MSK_IPAR_WRITE_INT_VARIABLES*
- *MSK_IPAR_WRITE_LP_FULL_OBJ*
- *MSK_SPAR_WRITE_LP_GEN_VAR_NAME*

- *MSK_IPAR_WRITE_LP_LINE_WIDTH*
- *MSK_IPAR_WRITE_LP_QUOTED_NAMES*
- *MSK_IPAR_WRITE_LP_STRICT_FORMAT*
- *MSK_IPAR_WRITE_LP_TERMS_PER_LINE*
- *MSK_IPAR_WRITE_MPS_FORMAT*
- *MSK_IPAR_WRITE_MPS_INT*
- *MSK_IPAR_WRITE_PRECISION*
- *MSK_IPAR_WRITE_SOL_BARVARIABLES*
- *MSK_IPAR_WRITE_SOL_CONSTRAINTS*
- *MSK_IPAR_WRITE_SOL_HEAD*
- *MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES*
- *MSK_IPAR_WRITE_SOL_VARIABLES*
- *MSK_IPAR_WRITE_TASK_INC_SOL*
- *MSK_IPAR_WRITE_XML_MODE*

10.2.20 Presolve parameters.

- *MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_FILL*
- *MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES*
- *MSK_IPAR_PRESOLVE_LEVEL*
- *MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH*
- *MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH*
- *MSK_IPAR_PRESOLVE_LINDEP_USE*
- *MSK_DPAR_PRESOLVE_TOL_ABS_LINDEP*
- *MSK_DPAR_PRESOLVE_TOL_AIJ*
- *MSK_DPAR_PRESOLVE_TOL_REL_LINDEP*
- *MSK_DPAR_PRESOLVE_TOL_S*
- *MSK_DPAR_PRESOLVE_TOL_X*
- *MSK_IPAR_PRESOLVE_USE*

10.2.21 Interior-point method parameters.

- *MSK_IPAR_BI_IGNORE_MAX_ITER*
- *MSK_IPAR_BI_IGNORE_NUM_ERROR*
- *MSK_DPAR_CHECK_CONVEXITY_REL_TOL*
- *MSK_IPAR_INTPNT_BASIS*
- *MSK_DPAR_INTPNT_CO_TOL_DFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_INFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_MU_RED*
- *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*

- *MSK_DPAR_INTPNT_CO_TOL_PFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_REL_GAP*
- *MSK_IPAR_INTPNT_DIFF_STEP*
- *MSK_IPAR_INTPNT_HOTSTART*
- *MSK_IPAR_INTPNT_MAX_ITERATIONS*
- *MSK_IPAR_INTPNT_MAX_NUM_COR*
- *MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS*
- *MSK_DPAR_INTPNT_NL_MERIT_BAL*
- *MSK_DPAR_INTPNT_NL_TOL_DFEAS*
- *MSK_DPAR_INTPNT_NL_TOL_MU_RED*
- *MSK_DPAR_INTPNT_NL_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_NL_TOL_PFEAS*
- *MSK_DPAR_INTPNT_NL_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_NL_TOL_REL_STEP*
- *MSK_IPAR_INTPNT_OFF_COL_TRH*
- *MSK_IPAR_INTPNT_ORDER_METHOD*
- *MSK_DPAR_INTPNT_QO_TOL_DFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_INFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_MU_RED*
- *MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_QO_TOL_PFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_REL_GAP*
- *MSK_IPAR_INTPNT_REGULARIZATION_USE*
- *MSK_IPAR_INTPNT_SCALING*
- *MSK_IPAR_INTPNT_SOLVE_FORM*
- *MSK_IPAR_INTPNT_STARTING_POINT*
- *MSK_DPAR_INTPNT_TOL_DFEAS*
- *MSK_DPAR_INTPNT_TOL_DSAFE*
- *MSK_DPAR_INTPNT_TOL_INFEAS*
- *MSK_DPAR_INTPNT_TOL_MU_RED*
- *MSK_DPAR_INTPNT_TOL_PATH*
- *MSK_DPAR_INTPNT_TOL_PFEAS*
- *MSK_DPAR_INTPNT_TOL_PSAFE*
- *MSK_DPAR_INTPNT_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_TOL_REL_STEP*
- *MSK_DPAR_INTPNT_TOL_STEP_SIZE*
- *MSK_IPAR_LOG_INTPNT*
- *MSK_IPAR_LOG_PRESOLVE*
- *MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL*

10.2.22 Analysis parameters.

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

10.2.23 Mixed-integer optimization parameters.

- *MSK_IPAR_LOG_MIO*
- *MSK_IPAR_LOG_MIO_FREQ*
- *MSK_IPAR_MIO_BRANCH_DIR*
- *MSK_IPAR_MIO_CONSTRUCT_SOL*
- *MSK_IPAR_MIO_CUT_CLIQUE*
- *MSK_IPAR_MIO_CUT_CMIR*
- *MSK_IPAR_MIO_CUT_GMI*
- *MSK_IPAR_MIO_CUT_IMPLIED_BOUND*
- *MSK_IPAR_MIO_CUT_KNAPSACK_COVER*
- *MSK_IPAR_MIO_CUT_SELECTION_LEVEL*
- *MSK_DPAR_MIO_DISABLE_TERM_TIME*
- *MSK_IPAR_MIO_HEURISTIC_LEVEL*
- *MSK_IPAR_MIO_MAX_NUM_BRANCHES*
- *MSK_IPAR_MIO_MAX_NUM_RELAXS*
- *MSK_IPAR_MIO_MAX_NUM_SOLUTIONS*
- *MSK_DPAR_MIO_MAX_TIME*
- *MSK_DPAR_MIO_NEAR_TOL_ABS_GAP*
- *MSK_DPAR_MIO_NEAR_TOL_REL_GAP*
- *MSK_IPAR_MIO_NODE_OPTIMIZER*
- *MSK_IPAR_MIO_NODE_SELECTION*
- *MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE*
- *MSK_IPAR_MIO_PROBING_LEVEL*
- *MSK_DPAR_MIO_REL_GAP_CONST*
- *MSK_IPAR_MIO_RINS_MAX_NODES*
- *MSK_IPAR_MIO_ROOT_OPTIMIZER*
- *MSK_IPAR_MIO_ROOT_REPEAT_PRESOLVE_LEVEL*
- *MSK_DPAR_MIO_TOL_ABS_GAP*
- *MSK_DPAR_MIO_TOL_ABS_RELAX_INT*
- *MSK_DPAR_MIO_TOL_FEAS*
- *MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT*
- *MSK_DPAR_MIO_TOL_REL_GAP*

- *MSK_IPAR_MIO_VB_DETECTION_LEVEL*

10.3 Response codes

- *Termination codes*
- *Error codes*
- *Warning codes*

10.3.1 Termination Codes

MSK_RES_OK (0)
: No error occurred.

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

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

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

MSK_RES_TRM_MAX_NUM_SETBACKS (10020)
: The optimizer terminated as the maximum number of set-backs was reached. This indicates % serious numerical problems and a possibly badly formulated problem.

MSK_RES_TRM_MAX_TIME (10001)
: The optimizer terminated at the maximum amount of time.

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

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

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

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

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

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

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

MSK_RES_TRM_STALL (10006)
: The optimizer is terminated due to slow progress.

  Stalling means that numerical problems prevent the optimizer from making reasonable progress and that it make no sense to continue. In many cases this happens if the problem is badly scaled or otherwise ill-conditioned. There is no guarantee that the solution will be (near) feasible or near optimal. However, often stalling happens near the optimum, and the returned solution may be of good quality. Therefore, it is recommended to check the status of then solution. If the solution near optimal the solution is most likely good enough for most practical purposes.

Please note that if a linear optimization problem is solved using the interior-point optimizer with basis identification turned on, the returned basic solution likely to have high accuracy, even though the optimizer stalled.

Some common causes of stalling are a) badly scaled models, b) near feasible or near infeasible problems and c) a non-convex problems. Case c) is only relevant for general non-linear problems. It is not possible in general for **MOSEK** to check if a specific problems is convex since such a check would be NP hard in itself. This implies that care should be taken when solving problems involving general user defined functions.

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

10.3.2 Error Codes

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

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

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

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

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

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

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

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

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

`MSK_RES_ERR_ARGUMENT_LENNEQ (1197)`
: Incorrect length of arguments.

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

`MSK_RES_ERR_ARGUMENT_TYPE (1198)`
: Incorrect argument type.

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

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

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

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

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

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

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

MSK_RES_ERR_CBF_DUPLICATE_ACOORD (7116)
Duplicate index in ACOORD.

MSK_RES_ERR_CBF_DUPLICATE_BCOORD (7115)
Duplicate index in BCOORD.

MSK_RES_ERR_CBF_DUPLICATE_CON (7108)
Duplicate CON keyword.

MSK_RES_ERR_CBF_DUPLICATE_INT (7110)
Duplicate INT keyword.

MSK_RES_ERR_CBF_DUPLICATE_OBJ (7107)
Duplicate OBJ keyword.

MSK_RES_ERR_CBF_DUPLICATE_OBJACOORD (7114)
Duplicate index in OBJCOORD.

MSK_RES_ERR_CBF_DUPLICATE_VAR (7109)
Duplicate VAR keyword.

MSK_RES_ERR_CBF_INVALID_CON_TYPE (7112)
Invalid constraint type.

MSK_RES_ERR_CBF_INVALID_DOMAIN_DIMENSION (7113)
Invalid domain dimension.

MSK_RES_ERR_CBF_INVALID_INT_INDEX (7121)
Invalid INT index.

MSK_RES_ERR_CBF_INVALID_VAR_TYPE (7111)
Invalid variable type.

MSK_RES_ERR_CBF_NO_VARIABLES (7102)
No variables are specified.

MSK_RES_ERR_CBF_NO_VERSION_SPECIFIED (7105)
No version specified.

MSK_RES_ERR_CBF_OBJ_SENSE (7101)
An invalid objective sense is specified.

MSK_RES_ERR_CBF_PARSE (7100)
An error occurred while parsing an CBF file.

MSK_RES_ERR_CBF_SYNTAX (7106)
Invalid syntax.

MSK_RES_ERR_CBF_TOO_FEW_CONSTRAINTS (7118)
Too few constraints defined.

MSK_RES_ERR_CBF_TOO_FEW_INTS (7119)
Too few ints are specified.

MSK_RES_ERR_CBF_TOO_FEW_VARIABLES (7117)
Too few variables defined.

MSK_RES_ERR_CBF_TOO_MANY_CONSTRAINTS (7103)
Too many constraints specified.

MSK_RES_ERR_CBF_TOO_MANY_INTS (7120)
Too many ints are specified.

`MSK_RES_ERR_CBF_TOO_MANY_VARIABLES (7104)`
Too many variables specified.

`MSK_RES_ERR_CBF_UNSUPPORTED (7122)`
Unsupported feature is present.

`MSK_RES_ERR_CON_Q_NOT_NSD (1294)`
The quadratic constraint matrix is not negative semidefinite as expected for a constraint with finite lower bound. This results in a nonconvex problem. The parameter *MSK_DPAR_CHECK_CONVEXITY_REL_TOL* can be used to relax the convexity check.

`MSK_RES_ERR_CON_Q_NOT_PSD (1293)`
The quadratic constraint matrix is not positive semidefinite as expected for a constraint with finite upper bound. This results in a nonconvex problem. The parameter *MSK_DPAR_CHECK_CONVEXITY_REL_TOL* can be used to relax the convexity check.

`MSK_RES_ERR_CONE_INDEX (1300)`
An index of a non-existing cone has been specified.

`MSK_RES_ERR_CONE_OVERLAP (1302)`
One or more of the variables in the cone to be added is already member of another cone. Now assume the variable is x_j then add a new variable say x_k and the constraint

$$x_j = x_k$$

and then let x_k be member of the cone to be appended.

`MSK_RES_ERR_CONE_OVERLAP_APPEND (1307)`
The cone to be appended has one variable which is already member of another cone.

`MSK_RES_ERR_CONE_REP_VAR (1303)`
A variable is included multiple times in the cone.

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

`MSK_RES_ERR_CONE_TYPE (1305)`
Invalid cone type specified.

`MSK_RES_ERR_CONE_TYPE_STR (1306)`
Invalid cone type specified.

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

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

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

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

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

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

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

`MSK_RES_ERR_END_OF_FILE (1059)`
End of file reached.

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

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

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

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

`MSK_RES_ERR_FILE_LICENSE (1007)`
Invalid license file.

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

`MSK_RES_ERR_FILE_READ (1053)`
File read error.

`MSK_RES_ERR_FILE_WRITE (1054)`
File write error.

`MSK_RES_ERR_FIRST (1261)`
Invalid `first`.

`MSK_RES_ERR_FIRSTI (1285)`
Invalid `firsti`.

`MSK_RES_ERR_FIRSTJ (1287)`
Invalid `firstj`.

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

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

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

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

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

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

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

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

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

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

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

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

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

MSK_RES_ERR_INF_DOU_NAME (1230)
A double information name is invalid.

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

MSK_RES_ERR_INF_INT_NAME (1231)
An integer information name is invalid.

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

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

MSK_RES_ERR_INF_TYPE (1232)
The information type is invalid.

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

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

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

MSK_RES_ERR_INTERNAL (3000)
An internal error occurred. Please report this problem.

MSK_RES_ERR_INTERNAL_TEST_FAILED (3500)
An internal unit test function failed.

MSK_RES_ERR_INV_APTRE (1253)
`aptre[j]` is strictly smaller than `aptrb[j]` for some `j`.

MSK_RES_ERR_INV_BK (1255)
Invalid bound key.

MSK_RES_ERR_INV_BKC (1256)
Invalid bound key is specified for a constraint.

MSK_RES_ERR_INV_BKX (1257)
An invalid bound key is specified for a variable.

MSK_RES_ERR_INV_CONE_TYPE (1272)
Invalid cone type code is encountered.

MSK_RES_ERR_INV_CONE_TYPE_STR (1271)
Invalid cone type string encountered.

MSK_RES_ERR_INV_MARKI (2501)
Invalid value in marki.

MSK_RES_ERR_INV_MARKJ (2502)
Invalid value in markj.

MSK_RES_ERR_INV_NAME_ITEM (1280)
An invalid name item code is used.

MSK_RES_ERR_INV_NUMI (2503)
Invalid numi.

MSK_RES_ERR_INV_NUMJ (2504)
Invalid numj.

MSK_RES_ERR_INV_OPTIMIZER (1550)
An invalid optimizer has been chosen for the problem. This means that the simplex or the conic optimizer is chosen to optimize a nonlinear problem.

MSK_RES_ERR_INV_PROBLEM (1500)
Invalid problem type. Probably a nonconvex problem has been specified.

MSK_RES_ERR_INV_QCON_SUBI (1405)
Invalid value in `qcsubi`.

MSK_RES_ERR_INV_QCON_SUBJ (1406)
Invalid value in `qcsubj`.

MSK_RES_ERR_INV_QCON_SUBK (1404)
Invalid value in `qcsubk`.

MSK_RES_ERR_INV_QCON_VAL (1407)
Invalid value in `qcval`.

MSK_RES_ERR_INV_QOBJ_SUBI (1401)
Invalid value in `qosubi`.

MSK_RES_ERR_INV_QOBJ_SUBJ (1402)
Invalid value in `qosubj`.

MSK_RES_ERR_INV_QOBJ_VAL (1403)
Invalid value in `qoval`.

MSK_RES_ERR_INV_SK (1270)
Invalid status key code.

MSK_RES_ERR_INV_SK_STR (1269)
Invalid status key string encountered.

MSK_RES_ERR_INV_SKC (1267)
Invalid value in `skc`.

MSK_RES_ERR_INV_SKN (1274)
Invalid value in `skn`.

MSK_RES_ERR_INV_SKX (1268)
Invalid value in `skx`.

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

MSK_RES_ERR_INVALID_ACCMODE (2520)
An invalid access mode is specified.

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

MSK_RES_ERR_INVALID_AMPL_STUB (3700)
Invalid AMPL stub.

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

MSK_RES_ERR_INVALID_COMPRESSION (1800)
Invalid compression type.

MSK_RES_ERR_INVALID_CON_NAME (1076)
An invalid constraint name is used.

MSK_RES_ERR_INVALID_CONE_NAME (1078)
An invalid cone name is used.

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

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

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

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

MSK_RES_ERR_INVALID_FORMAT_TYPE (1283)
Invalid format type.

MSK_RES_ERR_INVALID_IDX (1246)
A specified index is invalid.

MSK_RES_ERR_INVALID_IOMODE (1801)
Invalid io mode.

MSK_RES_ERR_INVALID_MAX_NUM (1247)
A specified index is invalid.

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

MSK_RES_ERR_INVALID_OBJ_NAME (1075)
An invalid objective name is specified.

MSK_RES_ERR_INVALID_OBJECTIVE_SENSE (1445)
An invalid objective sense is specified.

MSK_RES_ERR_INVALID_PROBLEM_TYPE (6000)
An invalid problem type.

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

MSK_RES_ERR_INVALID_STREAM (1062)
An invalid stream is referenced.

MSK_RES_ERR_INVALID_SURPLUS (1275)
Invalid surplus.

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

MSK_RES_ERR_INVALID_TASK (1064)
The `task` is invalid.

MSK_RES_ERR_INVALID_UTF8 (2900)
An invalid UTF8 string is encountered.

MSK_RES_ERR_INVALID_VAR_NAME (1077)
An invalid variable name is used.

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

MSK_RES_ERR_INVALID_WHICHSOL (1228)
`whichsol` is invalid.

MSK_RES_ERR_JSON_DATA (1179)
Inconsistent data in JSON Task file

MSK_RES_ERR_JSON_FORMAT (1178)
Error in an JSON Task file

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

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

MSK_RES_ERR_JSON_STRING (1176)
: Error in JSON string.

MSK_RES_ERR_JSON_SYNTAX (1175)
: Syntax error in an JSON data

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

MSK_RES_ERR_LASTI (1286)
: Invalid `lasti`.

MSK_RES_ERR_LASTJ (1288)
: Invalid `lastj`.

MSK_RES_ERR_LAU_ARG_K (7012)
: Invalid argument k.

MSK_RES_ERR_LAU_ARG_M (7010)
: Invalid argument m.

MSK_RES_ERR_LAU_ARG_N (7011)
: Invalid argument n.

MSK_RES_ERR_LAU_ARG_TRANS (7018)
: Invalid argument trans.

MSK_RES_ERR_LAU_ARG_TRANSA (7015)
: Invalid argument transa.

MSK_RES_ERR_LAU_ARG_TRANSB (7016)
: Invalid argument transb.

MSK_RES_ERR_LAU_ARG_UPLO (7017)
: Invalid argument uplo.

MSK_RES_ERR_LAU_INVALID_LOWER_TRIANGULAR_MATRIX (7002)
: An invalid lower triangular matrix.

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

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

MSK_RES_ERR_LAU_SINGULAR_MATRIX (7000)
: A matrix is singular.

MSK_RES_ERR_LAU_UNKNOWN (7005)
: An unknown error.

MSK_RES_ERR_LICENSE (1000)
: Invalid license.

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

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

MSK_RES_ERR_LICENSE_EXPIRED (1001)
: The license has expired.

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

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

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

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

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

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

- The feature has expired.
- The feature's start date is later than today's date.
- The version requested is higher than feature's the highest supported version.
- A corrupted license file.

Try restarting the license and inspect the license server debug file, usually called `lmgrd.log`.

`MSK_RES_ERR_LICENSE_SERVER (1015)`
The license server is not responding.

`MSK_RES_ERR_LICENSE_SERVER_VERSION (1026)`
The version specified in the checkout request is greater than the highest version number the daemon supports.

`MSK_RES_ERR_LICENSE_VERSION (1002)`
The license is valid for another version of **MOSEK**.

`MSK_RES_ERR_LINK_FILE_DLL (1040)`
A file cannot be linked to a stream in the DLL version.

`MSK_RES_ERR_LIVING_TASKS (1066)`
All tasks associated with an enviroment must be deleted before the environment is deleted. There are still some undeleted tasks.

`MSK_RES_ERR_LOWER_BOUND_IS_A_NAN (1390)`
The lower bound specified is not a number (nan).

`MSK_RES_ERR_LP_DUP_SLACK_NAME (1152)`
The name of the slack variable added to a ranged constraint already exists.

`MSK_RES_ERR_LP_EMPTY (1151)`
The problem cannot be written to an LP formatted file.

`MSK_RES_ERR_LP_FILE_FORMAT (1157)`
Syntax error in an LP file.

`MSK_RES_ERR_LP_FORMAT (1160)`
Syntax error in an LP file.

`MSK_RES_ERR_LP_FREE_CONSTRAINT (1155)`
Free constraints cannot be written in LP file format.

`MSK_RES_ERR_LP_INCOMPATIBLE (1150)`
The problem cannot be written to an LP formatted file.

`MSK_RES_ERR_LP_INVALID_CON_NAME (1171)`
A constraint name is invalid when used in an LP formatted file.

MSK_RES_ERR_LP_INVALID_VAR_NAME (1154)
A variable name is invalid when used in an LP formatted file.

MSK_RES_ERR_LP_WRITE_CONIC_PROBLEM (1163)
The problem contains cones that cannot be written to an LP formatted file.

MSK_RES_ERR_LP_WRITE_GECO_PROBLEM (1164)
The problem contains general convex terms that cannot be written to an LP formatted file.

MSK_RES_ERR_LU_MAX_NUM_TRIES (2800)
Could not compute the LU factors of the matrix within the maximum number of allowed tries.

MSK_RES_ERR_MAX_LEN_IS_TOO_SMALL (1289)
An maximum length that is too small has been specified.

MSK_RES_ERR_MAXNUMBARVAR (1242)
The maximum number of semidefinite variables specified is smaller than the number of semidefinite variables in the task.

MSK_RES_ERR_MAXNUMCON (1240)
The maximum number of constraints specified is smaller than the number of constraints in the task.

MSK_RES_ERR_MAXNUMCONE (1304)
The value specified for `maxnumcone` is too small.

MSK_RES_ERR_MAXNUMQNZ (1243)
The maximum number of non-zeros specified for the Q matrices is smaller than the number of non-zeros in the current Q matrices.

MSK_RES_ERR_MAXNUMVAR (1241)
The maximum number of variables specified is smaller than the number of variables in the task.

MSK_RES_ERR_MIO_INTERNAL (5010)
A fatal error occurred in the mixed integer optimizer. Please contact **MOSEK** support.

MSK_RES_ERR_MIO_INVALID_NODE_OPTIMIZER (7131)
An invalid node optimizer was selected for the problem type.

MSK_RES_ERR_MIO_INVALID_ROOT_OPTIMIZER (7130)
An invalid root optimizer was selected for the problem type.

MSK_RES_ERR_MIO_NO_OPTIMIZER (1551)
No optimizer is available for the current class of integer optimization problems.

MSK_RES_ERR_MIO_NOT_LOADED (1553)
The mixed-integer optimizer is not loaded.

MSK_RES_ERR_MISSING_LICENSE_FILE (1008)
MOSEK cannot license file or a token server. See the **MOSEK** installation manual for details.

MSK_RES_ERR_MIXED_CONIC_AND_NL (1501)
The problem contains nonlinear terms conic constraints. The requested operation cannot be applied to this type of problem.

MSK_RES_ERR_MPS_CONE_OVERLAP (1118)
A variable is specified to be a member of several cones.

MSK_RES_ERR_MPS_CONE_REPEAT (1119)
A variable is repeated within the `CSECTION`.

MSK_RES_ERR_MPS_CONE_TYPE (1117)
Invalid cone type specified in a `CSECTION`.

MSK_RES_ERR_MPS_DUPLICATE_Q_ELEMENT (1121)
Duplicate elements is specfied in a Q matrix.

MSK_RES_ERR_MPS_FILE (1100)
An error occurred while reading an MPS file.

MSK_RES_ERR_MPS_INV_BOUND_KEY (1108)
An invalid bound key occurred in an MPS file.

MSK_RES_ERR_MPS_INV_CON_KEY (1107)
An invalid constraint key occurred in an MPS file.

MSK_RES_ERR_MPS_INV_FIELD (1101)
A field in the MPS file is invalid. Probably it is too wide.

MSK_RES_ERR_MPS_INV_MARKER (1102)
An invalid marker has been specified in the MPS file.

MSK_RES_ERR_MPS_INV_SEC_NAME (1109)
An invalid section name occurred in an MPS file.

MSK_RES_ERR_MPS_INV_SEC_ORDER (1115)
The sections in the MPS data file are not in the correct order.

MSK_RES_ERR_MPS_INVALID_OBJ_NAME (1128)
An invalid objective name is specified.

MSK_RES_ERR_MPS_INVALID_OBJSENSE (1122)
An invalid objective sense is specified.

MSK_RES_ERR_MPS_MUL_CON_NAME (1112)
A constraint name was specified multiple times in the ROWS section.

MSK_RES_ERR_MPS_MUL_CSEC (1116)
Multiple CSECTIONs are given the same name.

MSK_RES_ERR_MPS_MUL_QOBJ (1114)
The Q term in the objective is specified multiple times in the MPS data file.

MSK_RES_ERR_MPS_MUL_QSEC (1113)
Multiple QSECTIONs are specified for a constraint in the MPS data file.

MSK_RES_ERR_MPS_NO_OBJECTIVE (1110)
No objective is defined in an MPS file.

MSK_RES_ERR_MPS_NON_SYMMETRIC_Q (1120)
A non symmetric matrice has been speciefied.

MSK_RES_ERR_MPS_NULL_CON_NAME (1103)
An empty constraint name is used in an MPS file.

MSK_RES_ERR_MPS_NULL_VAR_NAME (1104)
An empty variable name is used in an MPS file.

MSK_RES_ERR_MPS_SPLITTED_VAR (1111)
All elements in a column of the A matrix must be specified consecutively. Hence, it is illegal to specify non-zero elements in A for variable 1, then for variable 2 and then variable 1 again.

MSK_RES_ERR_MPS_TAB_IN_FIELD2 (1125)
A tab char occurred in field 2.

MSK_RES_ERR_MPS_TAB_IN_FIELD3 (1126)
A tab char occurred in field 3.

MSK_RES_ERR_MPS_TAB_IN_FIELD5 (1127)
A tab char occurred in field 5.

MSK_RES_ERR_MPS_UNDEF_CON_NAME (1105)
An undefined constraint name occurred in an MPS file.

MSK_RES_ERR_MPS_UNDEF_VAR_NAME (1106)
An undefined variable name occurred in an MPS file.

MSK_RES_ERR_MUL_A_ELEMENT (1254)
An element in A is defined multiple times.

`MSK_RES_ERR_NAME_IS_NULL (1760)`
The name buffer is a NULL pointer.

`MSK_RES_ERR_NAME_MAX_LEN (1750)`
A name is longer than the buffer that is supposed to hold it.

`MSK_RES_ERR_NAN_IN_BLC (1461)`
l^c contains an invalid floating point value, i.e. a `NaN`.

`MSK_RES_ERR_NAN_IN_BLX (1471)`
l^x contains an invalid floating point value, i.e. a `NaN`.

`MSK_RES_ERR_NAN_IN_BUC (1462)`
u^c contains an invalid floating point value, i.e. a `NaN`.

`MSK_RES_ERR_NAN_IN_BUX (1472)`
u^x contains an invalid floating point value, i.e. a `NaN`.

`MSK_RES_ERR_NAN_IN_C (1470)`
c contains an invalid floating point value, i.e. a `NaN`.

`MSK_RES_ERR_NAN_IN_DOUBLE_DATA (1450)`
An invalid floating point value was used in some double data.

`MSK_RES_ERR_NEGATIVE_APPEND (1264)`
Cannot append a negative number.

`MSK_RES_ERR_NEGATIVE_SURPLUS (1263)`
Negative surplus.

`MSK_RES_ERR_NEWER_DLL (1036)`
The dynamic link library is newer than the specified version.

`MSK_RES_ERR_NO_BARS_FOR_SOLUTION (3916)`
There is no $\bar{s}$ available for the solution specified. In particular note there are no $\bar{s}$ defined for the basic and integer solutions.

`MSK_RES_ERR_NO_BARX_FOR_SOLUTION (3915)`
There is no $\overline{X}$ available for the solution specified. In particular note there are no $\overline{X}$ defined for the basic and integer solutions.

`MSK_RES_ERR_NO_BASIS_SOL (1600)`
No basic solution is defined.

`MSK_RES_ERR_NO_DUAL_FOR_ITG_SOL (2950)`
No dual information is available for the integer solution.

`MSK_RES_ERR_NO_DUAL_INFEAS_CER (2001)`
A certificate of infeasibility is not available.

`MSK_RES_ERR_NO_INIT_ENV (1063)`
`env` is not initialized.

`MSK_RES_ERR_NO_OPTIMIZER_VAR_TYPE (1552)`
No optimizer is available for this class of optimization problems.

`MSK_RES_ERR_NO_PRIMAL_INFEAS_CER (2000)`
A certificate of primal infeasibility is not available.

`MSK_RES_ERR_NO_SNX_FOR_BAS_SOL (2953)`
s_n^x is not available for the basis solution.

`MSK_RES_ERR_NO_SOLUTION_IN_CALLBACK (2500)`
The required solution is not available.

`MSK_RES_ERR_NON_UNIQUE_ARRAY (5000)`
An array does not contain unique elements.

`MSK_RES_ERR_NONCONVEX (1291)`
The optimization problem is nonconvex.

`MSK_RES_ERR_NONLINEAR_EQUALITY (1290)`
The model contains a nonlinear equality which defines a nonconvex set.

`MSK_RES_ERR_NONLINEAR_FUNCTIONS_NOT_ALLOWED (1428)`
An operation that is invalid for problems with nonlinear functions defined has been attempted.

`MSK_RES_ERR_NONLINEAR_RANGED (1292)`
Nonlinear constraints with finite lower and upper bound always define a nonconvex feasible set.

`MSK_RES_ERR_NR_ARGUMENTS (1199)`
Incorrect number of function arguments.

`MSK_RES_ERR_NULL_ENV (1060)`
`env` is a NULL pointer.

`MSK_RES_ERR_NULL_POINTER (1065)`
An argument to a function is unexpectedly a NULL pointer.

`MSK_RES_ERR_NULL_TASK (1061)`
`task` is a NULL pointer.

`MSK_RES_ERR_NUMCONLIM (1250)`
Maximum number of constraints limit is exceeded.

`MSK_RES_ERR_NUMVARLIM (1251)`
Maximum number of variables limit is exceeded.

`MSK_RES_ERR_OBJ_Q_NOT_NSD (1296)`
The quadratic coefficient matrix in the objective is not negative semidefinite as expected for a maximization problem. The parameter *MSK_DPAR_CHECK_CONVEXITY_REL_TOL* can be used to relax the convexity check.

`MSK_RES_ERR_OBJ_Q_NOT_PSD (1295)`
The quadratic coefficient matrix in the objective is not positive semidefinite as expected for a minimization problem. The parameter *MSK_DPAR_CHECK_CONVEXITY_REL_TOL* can be used to relax the convexity check.

`MSK_RES_ERR_OBJECTIVE_RANGE (1260)`
Empty objective range.

`MSK_RES_ERR_OLDER_DLL (1035)`
The dynamic link library is older than the specified version.

`MSK_RES_ERR_OPEN_DL (1030)`
A dynamic link library could not be opened.

`MSK_RES_ERR_OPF_FORMAT (1168)`
Syntax error in an OPF file

`MSK_RES_ERR_OPF_NEW_VARIABLE (1169)`
Introducing new variables is now allowed. When a `[variables]` section is present, it is not allowed to introduce new variables later in the problem.

`MSK_RES_ERR_OPF_PREMATURE_EOF (1172)`
Premature end of file in an OPF file.

`MSK_RES_ERR_OPTIMIZER_LICENSE (1013)`
The optimizer required is not licensed.

`MSK_RES_ERR_OVERFLOW (1590)`
A computation produced an overflow i.e. a very large number.

`MSK_RES_ERR_PARAM_INDEX (1210)`
Parameter index is out of range.

MSK_RES_ERR_PARAM_IS_TOO_LARGE (1215)
The parameter value is too large.

MSK_RES_ERR_PARAM_IS_TOO_SMALL (1216)
The parameter value is too small.

MSK_RES_ERR_PARAM_NAME (1205)
The parameter name is not correct.

MSK_RES_ERR_PARAM_NAME_DOU (1206)
The parameter name is not correct for a double parameter.

MSK_RES_ERR_PARAM_NAME_INT (1207)
The parameter name is not correct for an integer parameter.

MSK_RES_ERR_PARAM_NAME_STR (1208)
The parameter name is not correct for a string parameter.

MSK_RES_ERR_PARAM_TYPE (1218)
The parameter type is invalid.

MSK_RES_ERR_PARAM_VALUE_STR (1217)
The parameter value string is incorrect.

MSK_RES_ERR_PLATFORM_NOT_LICENSED (1019)
A requested license feature is not available for the required platform.

MSK_RES_ERR_POSTSOLVE (1580)
An error occurred during the postsolve. Please contact **MOSEK** support.

MSK_RES_ERR_PRO_ITEM (1281)
An invalid problem is used.

MSK_RES_ERR_PROB_LICENSE (1006)
The software is not licensed to solve the problem.

MSK_RES_ERR_QCON_SUBI_TOO_LARGE (1409)
Invalid value in `qcsubi`.

MSK_RES_ERR_QCON_SUBI_TOO_SMALL (1408)
Invalid value in `qcsubi`.

MSK_RES_ERR_QCON_UPPER_TRIANGLE (1417)
An element in the upper triangle of a Q^k is specified. Only elements in the lower triangle should be specified.

MSK_RES_ERR_QOBJ_UPPER_TRIANGLE (1415)
An element in the upper triangle of Q^o is specified. Only elements in the lower triangle should be specified.

MSK_RES_ERR_READ_FORMAT (1090)
The specified format cannot be read.

MSK_RES_ERR_READ_LP_MISSING_END_TAG (1159)
Syntax error in LP file. Possibly missing End tag.

MSK_RES_ERR_READ_LP_NONEXISTING_NAME (1162)
A variable never occurred in objective or constraints.

MSK_RES_ERR_REMOVE_CONE_VARIABLE (1310)
A variable cannot be removed because it will make a cone invalid.

MSK_RES_ERR_REPAIR_INVALID_PROBLEM (1710)
The feasibility repair does not support the specified problem type.

MSK_RES_ERR_REPAIR_OPTIMIZATION_FAILED (1711)
Computation the optimal relaxation failed. The cause may have been numerical problems.

MSK_RES_ERR_SEN_BOUND_INVALID_LO (3054)
Analysis of lower bound requested for an index, where no lower bound exists.

MSK_RES_ERR_SEN_BOUND_INVALID_UP (3053)
Analysis of upper bound requested for an index, where no upper bound exists.

MSK_RES_ERR_SEN_FORMAT (3050)
Syntax error in sensitivity analysis file.

MSK_RES_ERR_SEN_INDEX_INVALID (3055)
Invalid range given in the sensitivity file.

MSK_RES_ERR_SEN_INDEX_RANGE (3052)
Index out of range in the sensitivity analysis file.

MSK_RES_ERR_SEN_INVALID_REGEXP (3056)
Syntax error in regexp or regexp longer than 1024.

MSK_RES_ERR_SEN_NUMERICAL (3058)
Numerical difficulties encountered performing the sensitivity analysis.

MSK_RES_ERR_SEN_SOLUTION_STATUS (3057)
No optimal solution found to the original problem given for sensitivity analysis.

MSK_RES_ERR_SEN_UNDEF_NAME (3051)
An undefined name was encountered in the sensitivity analysis file.

MSK_RES_ERR_SEN_UNHANDLED_PROBLEM_TYPE (3080)
Sensitivity analysis cannot be performed for the specified problem. Sensitivity analysis is only possible for linear problems.

MSK_RES_ERR_SERVER_CONNECT (8000)
Failed to connect to remote solver server. The server string or the port string were invalid, or the server did not accept connection.

MSK_RES_ERR_SERVER_PROTOCOL (8001)
Unexpected message or data from solver server.

MSK_RES_ERR_SERVER_STATUS (8002)
Server returned non-ok HTTP status code

MSK_RES_ERR_SERVER_TOKEN (8003)
The job ID specified is incorrect or invalid

MSK_RES_ERR_SIZE_LICENSE (1005)
The problem is bigger than the license.

MSK_RES_ERR_SIZE_LICENSE_CON (1010)
The problem has too many constraints to be solved with the available license.

MSK_RES_ERR_SIZE_LICENSE_INTVAR (1012)
The problem contains too many integer variables to be solved with the available license.

MSK_RES_ERR_SIZE_LICENSE_NUMCORES (3900)
The computer contains more cpu cores than the license allows for.

MSK_RES_ERR_SIZE_LICENSE_VAR (1011)
The problem has too many variables to be solved with the available license.

MSK_RES_ERR_SOL_FILE_INVALID_NUMBER (1350)
An invalid number is specified in a solution file.

MSK_RES_ERR_SOLITEM (1237)
The solution item number `solitem` is invalid. Please note that *MSK_SOL_ITEM_SNX* is invalid for the basic solution.

MSK_RES_ERR_SOLVER_PROBTYPE (1259)
Problem type does not match the chosen optimizer.

MSK_RES_ERR_SPACE (1051)
Out of space.

MSK_RES_ERR_SPACE_LEAKING (1080)
MOSEK is leaking memory. This can be due to either an incorrect use of **MOSEK** or a bug.

MSK_RES_ERR_SPACE_NO_INFO (1081)
No available information about the space usage.

MSK_RES_ERR_SYM_MAT_DUPLICATE (3944)
A value in a symmetric matric as been specified more than once.

MSK_RES_ERR_SYM_MAT_HUGE (1482)
A symmetric matrix contains a huge value in absolute size. The parameter *MSK_DPAR_DATA_SYM_MAT_TOL_HUGE* controls when an $e_{i,j}$ is considered huge.

MSK_RES_ERR_SYM_MAT_INVALID (1480)
A symmetric matrix contains an invalid floating point value, i.e. a NaN or an infinite value.

MSK_RES_ERR_SYM_MAT_INVALID_COL_INDEX (3941)
A column index specified for sparse symmetric matrix is invalid.

MSK_RES_ERR_SYM_MAT_INVALID_ROW_INDEX (3940)
A row index specified for sparse symmetric matrix is invalid.

MSK_RES_ERR_SYM_MAT_INVALID_VALUE (3943)
The numerical value specified in a sparse symmetric matrix is not a value floating value.

MSK_RES_ERR_SYM_MAT_NOT_LOWER_TRINGULAR (3942)
Only the lower triangular part of sparse symmetric matrix should be specified.

MSK_RES_ERR_TASK_INCOMPATIBLE (2560)
The Task file is incompatible with this platform. This results from reading a file on a 32 bit platform generated on a 64 bit platform.

MSK_RES_ERR_TASK_INVALID (2561)
The Task file is invalid.

MSK_RES_ERR_TASK_WRITE (2562)
Failed to write the task file.

MSK_RES_ERR_THREAD_COND_INIT (1049)
Could not initialize a condition.

MSK_RES_ERR_THREAD_CREATE (1048)
Could not create a thread. This error may occur if a large number of environments are created and not deleted again. In any case it is a good practice to minimize the number of environments created.

MSK_RES_ERR_THREAD_MUTEX_INIT (1045)
Could not initialize a mutex.

MSK_RES_ERR_THREAD_MUTEX_LOCK (1046)
Could not lock a mutex.

MSK_RES_ERR_THREAD_MUTEX_UNLOCK (1047)
Could not unlock a mutex.

MSK_RES_ERR_TOCONIC_CONSTR_NOT_CONIC (7153)
The constraint is not conic representable.

MSK_RES_ERR_TOCONIC_CONSTR_Q_NOT_PSD (7150)
The matrix defining the quadratric part of constraint is not positive semidefinite.

MSK_RES_ERR_TOCONIC_CONSTRAINT_FX (7151)
The quadratic constraint is an equality, thus not convex.

MSK_RES_ERR_TOCONIC_CONSTRAINT_RA (7152)
The quadratic constraint has finite lower and upper bound, and therefore it is not convex.

MSK_RES_ERR_TOCONIC_OBJECTIVE_NOT_PSD (7155)
The matrix defining the quadratric part of the objective function is not positive semidefinite.

MSK_RES_ERR_TOO_SMALL_MAX_NUM_NZ (1245)
The maximum number of non-zeros specified is too small.

MSK_RES_ERR_TOO_SMALL_MAXNUMANZ (1252)
The maximum number of non-zeros specified for A is smaller than the number of non-zeros in the current A.

MSK_RES_ERR_UNB_STEP_SIZE (3100)
A step size in an optimizer was unexpectedly unbounded. For instance, if the step-size becomes unbounded in phase 1 of the simplex algorithm then an error occurs. Normally this will happen only if the problem is badly formulated. Please contact **MOSEK** support if this error occurs.

MSK_RES_ERR_UNDEF_SOLUTION (1265)
MOSEK has the following solution types:

- an interior-point solution,
- an basic solution,
- and an integer solution.

Each optimizer may set one or more of these solutions; e.g by default a successful optimization with the interior-point optimizer defines the interior-point solution, and, for linear problems, also the basic solution. This error occurs when asking for a solution or for information about a solution that is not defined.

MSK_RES_ERR_UNDEFINED_OBJECTIVE_SENSE (1446)
The objective sense has not been specified before the optimization.

MSK_RES_ERR_UNHANDLED_SOLUTION_STATUS (6010)
Unhandled solution status.

MSK_RES_ERR_UNKNOWN (1050)
Unknown error.

MSK_RES_ERR_UPPER_BOUND_IS_A_NAN (1391)
The upper bound specified is not a number (nan).

MSK_RES_ERR_UPPER_TRIANGLE (6020)
An element in the upper triangle of a lower triangular matrix is specified.

MSK_RES_ERR_USER_FUNC_RET (1430)
An user function reported an error.

MSK_RES_ERR_USER_FUNC_RET_DATA (1431)
An user function returned invalid data.

MSK_RES_ERR_USER_NLO_EVAL (1433)
The user-defined nonlinear function reported an error.

MSK_RES_ERR_USER_NLO_EVAL_HESSUBI (1440)
The user-defined nonlinear function reported an invalid subscript in the Hessian.

MSK_RES_ERR_USER_NLO_EVAL_HESSUBJ (1441)
The user-defined nonlinear function reported an invalid subscript in the Hessian.

MSK_RES_ERR_USER_NLO_FUNC (1432)
The user-defined nonlinear function reported an error.

MSK_RES_ERR_WHICHITEM_NOT_ALLOWED (1238)
whichitem is unacceptable.

MSK_RES_ERR_WHICHSOL (1236)
The solution defined by `whichsol` does not exists.

MSK_RES_ERR_WRITE_LP_FORMAT (1158)
Problem cannot be written as an LP file.

MSK_RES_ERR_WRITE_LP_NON_UNIQUE_NAME (1161)
An auto-generated name is not unique.

MSK_RES_ERR_WRITE_MPS_INVALID_NAME (1153)
An invalid name is created while writing an MPS file. Usually this will make the MPS file unreadable.

MSK_RES_ERR_WRITE_OPF_INVALID_VAR_NAME (1156)
Empty variable names cannot be written to OPF files.

MSK_RES_ERR_WRITING_FILE (1166)
An error occurred while writing file

MSK_RES_ERR_XML_INVALID_PROBLEM_TYPE (3600)
The problem type is not supported by the XML format.

MSK_RES_ERR_Y_IS_UNDEFINED (1449)
The solution item y is undefined.

10.3.3 Warning Codes

MSK_RES_WRN_ANA_ALMOST_INT_BOUNDS (904)
This warning is issued by the problem analyzer if a constraint is bound nearly integral.

MSK_RES_WRN_ANA_C_ZERO (901)
This warning is issued by the problem analyzer, if the coefficients in the linear part of the objective are all zero.

MSK_RES_WRN_ANA_CLOSE_BOUNDS (903)
This warning is issued by problem analyzer, if ranged constraints or variables with very close upper and lower bounds are detected. One should consider treating such constraints as equalities and such variables as constants.

MSK_RES_WRN_ANA_EMPTY_COLS (902)
This warning is issued by the problem analyzer, if columns, in which all coefficients are zero, are found.

MSK_RES_WRN_ANA_LARGE_BOUNDS (900)
This warning is issued by the problem analyzer, if one or more constraint or variable bounds are very large. One should consider omitting these bounds entirely by setting them to +inf or -inf.

MSK_RES_WRN_CONSTRUCT_INVALID_SOL_ITG (807)
The initial value for one or more of the integer variables is not feasible.

MSK_RES_WRN_CONSTRUCT_NO_SOL_ITG (810)
The construct solution requires an integer solution.

MSK_RES_WRN_CONSTRUCT_SOLUTION_INFEAS (805)
After fixing the integer variables at the suggested values then the problem is infeasible.

MSK_RES_WRN_DROPPED_NZ_QOBJ (201)
One or more non-zero elements were dropped in the Q matrix in the objective.

MSK_RES_WRN_DUPLICATE_BARVARIABLE_NAMES (852)
Two barvariable names are identical.

MSK_RES_WRN_DUPLICATE_CONE_NAMES (853)
Two cone names are identical.

MSK_RES_WRN_DUPLICATE_CONSTRAINT_NAMES (850)
Two constraint names are identical.

MSK_RES_WRN_DUPLICATE_VARIABLE_NAMES (851)
Two variable names are identical.

MSK_RES_WRN_ELIMINATOR_SPACE (801)
The eliminator is skipped at least once due to lack of space.

MSK_RES_WRN_EMPTY_NAME (502)
A variable or constraint name is empty. The output file may be invalid.

MSK_RES_WRN_IGNORE_INTEGER (250)
Ignored integer constraints.

MSK_RES_WRN_INCOMPLETE_LINEAR_DEPENDENCY_CHECK (800)
The linear dependency check(s) is incomplete. Normally this is not an important warning unless the optimization problem has been formulated with linear dependencies. Linear dependencies may prevent **MOSEK** from solving the problem.

MSK_RES_WRN_LARGE_AIJ (62)
A numerically large value is specified for an $a_{i,j}$ element in A. The parameter *MSK_DPAR_DATA_TOL_AIJ_LARGE* controls when an $a_{i,j}$ is considered large.

MSK_RES_WRN_LARGE_BOUND (51)
A numerically large bound value is specified.

MSK_RES_WRN_LARGE_CJ (57)
A numerically large value is specified for one c_j.

MSK_RES_WRN_LARGE_CON_FX (54)
An equality constraint is fixed to a numerically large value. This can cause numerical problems.

MSK_RES_WRN_LARGE_LO_BOUND (52)
A numerically large lower bound value is specified.

MSK_RES_WRN_LARGE_UP_BOUND (53)
A numerically large upper bound value is specified.

MSK_RES_WRN_LICENSE_EXPIRE (500)
The license expires.

MSK_RES_WRN_LICENSE_FEATURE_EXPIRE (505)
The license expires.

MSK_RES_WRN_LICENSE_SERVER (501)
The license server is not responding.

MSK_RES_WRN_LP_DROP_VARIABLE (85)
Ignored a variable because the variable was not previously defined. Usually this implies that a variable appears in the bound section but not in the objective or the constraints.

MSK_RES_WRN_LP_OLD_QUAD_FORMAT (80)
Missing '/2' after quadratic expressions in bound or objective.

MSK_RES_WRN_MIO_INFEASIBLE_FINAL (270)
The final mixed-integer problem with all the integer variables fixed at their optimal values is infeasible.

MSK_RES_WRN_MPS_SPLIT_BOU_VECTOR (72)
A BOUNDS vector is split into several nonadjacent parts in an MPS file.

MSK_RES_WRN_MPS_SPLIT_RAN_VECTOR (71)
A RANGE vector is split into several nonadjacent parts in an MPS file.

MSK_RES_WRN_MPS_SPLIT_RHS_VECTOR (70)
An RHS vector is split into several nonadjacent parts in an MPS file.

MSK_RES_WRN_NAME_MAX_LEN (65)
A name is longer than the buffer that is supposed to hold it.

MSK_RES_WRN_NO_DUALIZER (950)
No automatic dualizer is available for the specified problem. The primal problem is solved.

MSK_RES_WRN_NO_GLOBAL_OPTIMIZER (251)
No global optimizer is available.

MSK_RES_WRN_NO_NONLINEAR_FUNCTION_WRITE (450)
The problem contains a general nonlinear function in either the objective or the constraints. Such a nonlinear function cannot be written to a disk file. Note that quadratic terms when inputted explicitly can be written to disk.

MSK_RES_WRN_NZ_IN_UPR_TRI (200)
Non-zero elements specified in the upper triangle of a matrix were ignored.

MSK_RES_WRN_OPEN_PARAM_FILE (50)
The parameter file could not be opened.

MSK_RES_WRN_PARAM_IGNORED_CMIO (516)
A parameter was ignored by the conic mixed integer optimizer.

MSK_RES_WRN_PARAM_NAME_DOU (510)
The parameter name is not recognized as a double parameter.

MSK_RES_WRN_PARAM_NAME_INT (511)
The parameter name is not recognized as a integer parameter.

MSK_RES_WRN_PARAM_NAME_STR (512)
The parameter name is not recognized as a string parameter.

MSK_RES_WRN_PARAM_STR_VALUE (515)
The string is not recognized as a symbolic value for the parameter.

MSK_RES_WRN_PRESOLVE_OUTOFSPACE (802)
The presolve is incomplete due to lack of space.

MSK_RES_WRN_QUAD_CONES_WITH_ROOT_FIXED_AT_ZERO (930)
For at least one quadratic cone the root is fixed at (nearly) zero. This may cause problems such as a very large dual solution. Therefore, it is recommended to remove such cones before optimizing the problems, or to fix all the variables in the cone to 0.

MSK_RES_WRN_RQUAD_CONES_WITH_ROOT_FIXED_AT_ZERO (931)
For at least one rotated quadratic cone at least one of the root variables are fixed at (nearly) zero. This may cause problems such as a very large dual solution. Therefore, it is recommended to remove such cones before optimizing the problems, or to fix all the variables in the cone to 0.

MSK_RES_WRN_SOL_FILE_IGNORED_CON (351)
One or more lines in the constraint section were ignored when reading a solution file.

MSK_RES_WRN_SOL_FILE_IGNORED_VAR (352)
One or more lines in the variable section were ignored when reading a solution file.

MSK_RES_WRN_SOL_FILTER (300)
Invalid solution filter is specified.

MSK_RES_WRN_SPAR_MAX_LEN (66)
A value for a string parameter is longer than the buffer that is supposed to hold it.

MSK_RES_WRN_SYM_MAT_LARGE (960)
A numerically large value is specified for an $e_{i,j}$ element in E. The parameter *MSK_DPAR_DATA_SYM_MAT_TOL_LARGE* controls when an $e_{i,j}$ is considered large.

MSK_RES_WRN_TOO_FEW_BASIS_VARS (400)
An incomplete basis has been specified. Too few basis variables are specified.

MSK_RES_WRN_TOO_MANY_BASIS_VARS (405)
A basis with too many variables has been specified.

MSK_RES_WRN_UNDEF_SOL_FILE_NAME (350)
Undefined name occurred in a solution.

MSK_RES_WRN_USING_GENERIC_NAMES (503)
Generic names are used because a name is not valid. For instance when writing an LP file the names must not contain blanks or start with a digit.

MSK_RES_WRN_WRITE_CHANGED_NAMES (803)
Some names were changed because they were invalid for the output file format.

MSK_RES_WRN_WRITE_DISCARDED_CFIX (804)
The fixed objective term could not be converted to a variable and was discarded in the output file.

MSK_RES_WRN_ZERO_AIJ (63)
One or more zero elements are specified in A.

MSK_RES_WRN_ZEROS_IN_SPARSE_COL (710)
One or more (near) zero elements are specified in a sparse column of a matrix. It is redundant to specify zero elements. Hence, it may indicate an error.

MSK_RES_WRN_ZEROS_IN_SPARSE_ROW (705)
One or more (near) zero elements are specified in a sparse row of a matrix. Since, it is redundant to specify zero elements then it may indicate an error.

10.4 Enumerations

MSKlanguagee
Language selection constants

MSK_LANG_ENG
English language selection

MSK_LANG_DAN
Danish language selection

MSKaccmodee
Constraint or variable access modes

MSK_ACC_VAR
Access data by columns (variable oriented)

MSK_ACC_CON
Access data by rows (constraint oriented)

MSKbasindtypee
Basis identification

MSK_BI_NEVER
Never do basis identification.

MSK_BI_ALWAYS
Basis identification is always performed even if the interior-point optimizer terminates abnormally.

MSK_BI_NO_ERROR
Basis identification is performed if the interior-point optimizer terminates without an error.

MSK_BI_IF_FEASIBLE
Basis identification is not performed if the interior-point optimizer terminates with a problem status saying that the problem is primal or dual infeasible.

MSK_BI_RESERVERED
Not currently in use.

MSKboundkeye
Bound keys

MSK_BK_LO
The constraint or variable has a finite lower bound and an infinite upper bound.

MSK_BK_UP
The constraint or variable has an infinite lower bound and an finite upper bound.

MSK_BK_FX
The constraint or variable is fixed.

MSK_BK_FR
The constraint or variable is free.

MSK_BK_RA
The constraint or variable is ranged.

MSKmarke
Mark

MSK_MARK_LO
The lower bound is selected for sensitivity analysis.

MSK_MARK_UP
The upper bound is selected for sensitivity analysis.

MSKsimdegene
Degeneracy strategies

MSK_SIM_DEGEN_NONE
The simplex optimizer should use no degeneration strategy.

MSK_SIM_DEGEN_FREE
The simplex optimizer chooses the degeneration strategy.

MSK_SIM_DEGEN_AGGRESSIVE
The simplex optimizer should use an aggressive degeneration strategy.

MSK_SIM_DEGEN_MODERATE
The simplex optimizer should use a moderate degeneration strategy.

MSK_SIM_DEGEN_MINIMUM
The simplex optimizer should use a minimum degeneration strategy.

MSKtransposee
Transposed matrix.

MSK_TRANSPOSE_NO
No transpose is applied.

MSK_TRANSPOSE_YES
A transpose is applied.

MSKuploe
Triangular part of a symmetric matrix.

MSK_UPLO_LO
Lower part.

MSK_UPLO_UP
Upper part

MSKsimreforme
Problem reformulation.

MSK_SIM_REFORMULATION_ON
Allow the simplex optimizer to reformulate the problem.

MSK_SIM_REFORMULATION_OFF
Disallow the simplex optimizer to reformulate the problem.

MSK_SIM_REFORMULATION_FREE
The simplex optimizer can choose freely.

MSK_SIM_REFORMULATION_AGGRESSIVE
The simplex optimizer should use an aggressive reformulation strategy.

MSKsimdupvece
Exploit duplicate columns.

MSK_SIM_EXPLOIT_DUPVEC_ON
Allow the simplex optimizer to exploit duplicated columns.

MSK_SIM_EXPLOIT_DUPVEC_OFF
Disallow the simplex optimizer to exploit duplicated columns.

MSK_SIM_EXPLOIT_DUPVEC_FREE
The simplex optimizer can choose freely.

MSKsimhotstarte
Hot-start type employed by the simplex optimizer

MSK_SIM_HOTSTART_NONE
The simplex optimizer performs a coldstart.

MSK_SIM_HOTSTART_FREE
The simplex optimize chooses the hot-start type.

MSK_SIM_HOTSTART_STATUS_KEYS
Only the status keys of the constraints and variables are used to choose the type of hot-start.

MSKintpnthotstarte
Hot-start type employed by the interior-point optimizers.

MSK_INTPNT_HOTSTART_NONE
The interior-point optimizer performs a coldstart.

MSK_INTPNT_HOTSTART_PRIMAL
The interior-point optimizer exploits the primal solution only.

MSK_INTPNT_HOTSTART_DUAL
The interior-point optimizer exploits the dual solution only.

MSK_INTPNT_HOTSTART_PRIMAL_DUAL
The interior-point optimizer exploits both the primal and dual solution.

MSKcallbackcodee
Progress call-back codes

MSK_CALLBACK_BEGIN_ROOT_CUTGEN
The call-back function is called when root cut generation is started.

MSK_CALLBACK_IM_ROOT_CUTGEN
The call-back is called from within root cut generation at an intermediate stage.

MSK_CALLBACK_END_ROOT_CUTGEN
The call-back function is called when root cut generation is is terminated.

MSK_CALLBACK_BEGIN_OPTIMIZER
The call-back function is called when the optimizer is started.

MSK_CALLBACK_END_OPTIMIZER
The call-back function is called when the optimizer is terminated.

MSK_CALLBACK_BEGIN_PRESOLVE
The call-back function is called when the presolve is started.

MSK_CALLBACK_UPDATE_PRESOLVE
The call-back function is called from within the presolve procedure.

MSK_CALLBACK_IM_PRESOLVE
The call-back function is called from within the presolve procedure at an intermediate stage.

MSK_CALLBACK_END_PRESOLVE
The call-back function is called when the presolve is completed.

MSK_CALLBACK_BEGIN_INTPNT
The call-back function is called when the interior-point optimizer is started.

MSK_CALLBACK_INTPNT
The call-back function is called from within the interior-point optimizer after the information database has been updated.

MSK_CALLBACK_IM_INTPNT
The call-back function is called at an intermediate stage within the interior-point optimizer where the information database has not been updated.

MSK_CALLBACK_END_INTPNT
The call-back function is called when the interior-point optimizer is terminated.

MSK_CALLBACK_BEGIN_CONIC
The call-back function is called when the conic optimizer is started.

MSK_CALLBACK_CONIC
The call-back function is called from within the conic optimizer after the information database has been updated.

MSK_CALLBACK_IM_CONIC
The call-back function is called at an intermediate stage within the conic optimizer where the information database has not been updated.

MSK_CALLBACK_END_CONIC
The call-back function is called when the conic optimizer is terminated.

MSK_CALLBACK_PRIMAL_SIMPLEX
The call-back function is called from within the primal simplex optimizer.

MSK_CALLBACK_DUAL_SIMPLEX
The call-back function is called from within the dual simplex optimizer.

MSK_CALLBACK_BEGIN_BI
The basis identification procedure has been started.

MSK_CALLBACK_IM_BI
The call-back function is called from within the basis identification procedure at an intermediate point.

MSK_CALLBACK_END_BI
The call-back function is called when the basis identification procedure is terminated.

MSK_CALLBACK_BEGIN_PRIMAL_BI
The call-back function is called from within the basis identification procedure when the primal phase is started.

MSK_CALLBACK_IM_PRIMAL_BI
The call-back function is called from within the basis identification procedure at an intermediate point in the primal phase.

MSK_CALLBACK_UPDATE_PRIMAL_BI
The call-back function is called from within the basis identification procedure at an intermediate point in the primal phase.

MSK_CALLBACK_END_PRIMAL_BI
The call-back function is called from within the basis identification procedure when the primal phase is terminated.

MSK_CALLBACK_BEGIN_DUAL_BI
: The call-back function is called from within the basis identification procedure when the dual phase is started.

MSK_CALLBACK_IM_DUAL_BI
: The call-back function is called from within the basis identification procedure at an intermediate point in the dual phase.

MSK_CALLBACK_UPDATE_DUAL_BI
: The call-back function is called from within the basis identification procedure at an intermediate point in the dual phase.

MSK_CALLBACK_END_DUAL_BI
: The call-back function is called from within the basis identification procedure when the dual phase is terminated.

MSK_CALLBACK_BEGIN_SIMPLEX_BI
: The call-back function is called from within the basis identification procedure when the simplex clean-up phase is started.

MSK_CALLBACK_IM_SIMPLEX_BI
: The call-back function is called from within the basis identification procedure at an intermediate point in the simplex clean-up phase. The frequency of the call-backs is controlled by the *MSK_IPAR_LOG_SIM_FREQ* parameter.

MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX_BI
: The call-back function is called from within the basis identification procedure when the primal simplex clean-up phase is started.

MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX_BI
: The call-back function is called from within the basis identification procedure at an intermediate point in the primal simplex clean-up phase. The frequency of the call-backs is controlled by the *MSK_IPAR_LOG_SIM_FREQ* parameter.

MSK_CALLBACK_END_PRIMAL_SIMPLEX_BI
: The call-back function is called from within the basis identification procedure when the primal clean-up phase is terminated.

MSK_CALLBACK_BEGIN_PRIMAL_DUAL_SIMPLEX_BI
: The call-back function is called from within the basis identification procedure when the primal-dual simplex clean-up phase is started.

MSK_CALLBACK_UPDATE_PRIMAL_DUAL_SIMPLEX_BI
: The call-back function is called from within the basis identification procedure at an intermediate point in the primal-dual simplex clean-up phase. The frequency of the call-backs is controlled by the *MSK_IPAR_LOG_SIM_FREQ* parameter.

MSK_CALLBACK_END_PRIMAL_DUAL_SIMPLEX_BI
: The call-back function is called from within the basis identification procedure when the primal-dual clean-up phase is terminated.

MSK_CALLBACK_BEGIN_DUAL_SIMPLEX_BI
: The call-back function is called from within the basis identification procedure when the dual simplex clean-up phase is started.

MSK_CALLBACK_UPDATE_DUAL_SIMPLEX_BI
: The call-back function is called from within the basis identification procedure at an intermediate point in the dual simplex clean-up phase. The frequency of the call-backs is controlled by the *MSK_IPAR_LOG_SIM_FREQ* parameter.

MSK_CALLBACK_END_DUAL_SIMPLEX_BI
: The call-back function is called from within the basis identification procedure when the dual clean-up phase is terminated.

MSK_CALLBACK_END_SIMPLEX_BI
: The call-back function is called from within the basis identification procedure when the simplex clean-up phase is terminated.

MSK_CALLBACK_BEGIN_MIO
: The call-back function is called when the mixed-integer optimizer is started.

MSK_CALLBACK_IM_MIO
: The call-back function is called at an intermediate point in the mixed-integer optimizer.

MSK_CALLBACK_NEW_INT_MIO
: The call-back function is called after a new integer solution has been located by the mixed-integer optimizer.

MSK_CALLBACK_END_MIO
: The call-back function is called when the mixed-integer optimizer is terminated.

MSK_CALLBACK_BEGIN_SIMPLEX
: The call-back function is called when the simplex optimizer is started.

MSK_CALLBACK_BEGIN_DUAL_SIMPLEX
: The call-back function is called when the dual simplex optimizer started.

MSK_CALLBACK_IM_DUAL_SIMPLEX
: The call-back function is called at an intermediate point in the dual simplex optimizer.

MSK_CALLBACK_UPDATE_DUAL_SIMPLEX
: The call-back function is called in the dual simplex optimizer.

MSK_CALLBACK_END_DUAL_SIMPLEX
: The call-back function is called when the dual simplex optimizer is terminated.

MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX
: The call-back function is called when the primal simplex optimizer is started.

MSK_CALLBACK_IM_PRIMAL_SIMPLEX
: The call-back function is called at an intermediate point in the primal simplex optimizer.

MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX
: The call-back function is called in the primal simplex optimizer.

MSK_CALLBACK_END_PRIMAL_SIMPLEX
: The call-back function is called when the primal simplex optimizer is terminated.

MSK_CALLBACK_BEGIN_PRIMAL_DUAL_SIMPLEX
: The call-back function is called when the primal-dual simplex optimizer is started.

MSK_CALLBACK_IM_PRIMAL_DUAL_SIMPLEX
: The call-back function is called at an intermediate point in the primal-dual simplex optimizer.

MSK_CALLBACK_UPDATE_PRIMAL_DUAL_SIMPLEX
: The call-back function is called in the primal-dual simplex optimizer.

MSK_CALLBACK_END_PRIMAL_DUAL_SIMPLEX
: The call-back function is called when the primal-dual simplex optimizer is terminated.

MSK_CALLBACK_END_SIMPLEX
: The call-back function is called when the simplex optimizer is terminated.

MSK_CALLBACK_BEGIN_INFEAS_ANA
: The call-back function is called when the infeasibility analyzer is started.

MSK_CALLBACK_END_INFEAS_ANA
: The call-back function is called when the infeasibility analyzer is terminated.

MSK_CALLBACK_IM_PRIMAL_SENSIVITY
: The call-back function is called at an intermediate stage of the primal sensitivity analysis.

MSK_CALLBACK_IM_DUAL_SENSIVITY
: The call-back function is called at an intermediate stage of the dual sensitivity analysis.

MSK_CALLBACK_IM_MIO_INTPNT
: The call-back function is called at an intermediate point in the mixed-integer optimizer while running the interior-point optimizer.

MSK_CALLBACK_IM_MIO_PRIMAL_SIMPLEX
: The call-back function is called at an intermediate point in the mixed-integer optimizer while running the primal simplex optimizer.

MSK_CALLBACK_IM_MIO_DUAL_SIMPLEX
: The call-back function is called at an intermediate point in the mixed-integer optimizer while running the dual simplex optimizer.

MSK_CALLBACK_BEGIN_PRIMAL_SETUP_BI
: The call-back function is called when the primal BI setup is started.

MSK_CALLBACK_END_PRIMAL_SETUP_BI
: The call-back function is called when the primal BI setup is terminated.

MSK_CALLBACK_BEGIN_DUAL_SETUP_BI
: The call-back function is called when the dual BI phase is started.

MSK_CALLBACK_END_DUAL_SETUP_BI
: The call-back function is called when the dual BI phase is terminated.

MSK_CALLBACK_BEGIN_PRIMAL_SENSITIVITY
: Primal sensitivity analysis is started.

MSK_CALLBACK_END_PRIMAL_SENSITIVITY
: Primal sensitivity analysis is terminated.

MSK_CALLBACK_BEGIN_DUAL_SENSITIVITY
: Dual sensitivity analysis is started.

MSK_CALLBACK_END_DUAL_SENSITIVITY
: Dual sensitivity analysis is terminated.

MSK_CALLBACK_BEGIN_LICENSE_WAIT
: Begin waiting for license.

MSK_CALLBACK_END_LICENSE_WAIT
: End waiting for license.

MSK_CALLBACK_IM_LICENSE_WAIT
: **MOSEK** is waiting for a license.

MSK_CALLBACK_BEGIN_QCQO_REFORMULATE
: Begin QCQO reformulation.

MSK_CALLBACK_END_QCQO_REFORMULATE
: End QCQO reformulation.

MSK_CALLBACK_IM_QO_REFORMULATE
: The call-back function is called at an intermediate stage of the conic quadratic reformulation.

MSK_CALLBACK_BEGIN_TO_CONIC
: Begin conic reformulation.

MSK_CALLBACK_END_TO_CONIC
: End conic reformulation.

MSK_CALLBACK_BEGIN_FULL_CONVEXITY_CHECK
: Begin full convexity check.

MSK_CALLBACK_END_FULL_CONVEXITY_CHECK
: End full convexity check.

MSK_CALLBACK_IM_FULL_CONVEXITY_CHECK
The call-back function is called at an intermediate stage of the full convexity check.

MSK_CALLBACK_BEGIN_PRIMAL_REPAIR
Begin primal feasibility repair.

MSK_CALLBACK_END_PRIMAL_REPAIR
End primal feasibility repair.

MSK_CALLBACK_BEGIN_READ
MOSEK has started reading a problem file.

MSK_CALLBACK_IM_READ
Intermediate stage in reading.

MSK_CALLBACK_END_READ
MOSEK has finished reading a problem file.

MSK_CALLBACK_BEGIN_WRITE
MOSEK has started writing a problem file.

MSK_CALLBACK_END_WRITE
MOSEK has finished writing a problem file.

MSK_CALLBACK_READ_OPF_SECTION
A chunk of Q non-zeros has been read from a problem file.

MSK_CALLBACK_IM_LU
The call-back function is called from within the LU factorization procedure at an intermediate point.

MSK_CALLBACK_IM_ORDER
The call-back function is called from within the matrix ordering procedure at an intermediate point.

MSK_CALLBACK_IM_SIMPLEX
The call-back function is called from within the simplex optimizer at an intermediate point.

MSK_CALLBACK_READ_OPF
The call-back function is called from the OPF reader.

MSK_CALLBACK_WRITE_OPF
The call-back function is called from the OPF writer.

MSK_CALLBACK_SOLVING_REMOTE
The call-back function is called while the task is being solved on a remote server.

MSKcheckconvexitytypee
Types of convexity checks.

MSK_CHECK_CONVEXITY_NONE
No convexity check.

MSK_CHECK_CONVEXITY_SIMPLE
Perform simple and fast convexity check.

MSK_CHECK_CONVEXITY_FULL
Perform a full convexity check.

MSKcompresstypee
Compression types

MSK_COMPRESS_NONE
No compression is used.

MSK_COMPRESS_FREE
The type of compression used is chosen automatically.

MSK_COMPRESS_GZIP
The type of compression used is gzip compatible.

MSKconetypee
Cone types

MSK_CT_QUAD
The cone is a quadratic cone.

MSK_CT_RQUAD
The cone is a rotated quadratic cone.

MSKnametypee
Name types

MSK_NAME_TYPE_GEN
General names. However, no duplicate and blank names are allowed.

MSK_NAME_TYPE_MPS
MPS type names.

MSK_NAME_TYPE_LP
LP type names.

MSKsymmattypee
Cone types

MSK_SYMMAT_TYPE_SPARSE
Sparse symmetric matrix.

MSKdataformate
Data format types

MSK_DATA_FORMAT_EXTENSION
The file extension is used to determine the data file format.

MSK_DATA_FORMAT_MPS
The data file is MPS formatted.

MSK_DATA_FORMAT_LP
The data file is LP formatted.

MSK_DATA_FORMAT_OP
The data file is an optimization problem formatted file.

MSK_DATA_FORMAT_XML
The data file is an XML formatted file.

MSK_DATA_FORMAT_FREE_MPS
The data a free MPS formatted file.

MSK_DATA_FORMAT_TASK
Generic task dump file.

MSK_DATA_FORMAT_CB
Conic benchmark format,

MSK_DATA_FORMAT_JSON_TASK
JSON based task format.

MSKdinfiteme
Double information items

MSK_DINF_BI_TIME
Time spent within the basis identification procedure since its invocation.

MSK_DINF_BI_PRIMAL_TIME
Time spent within the primal phase of the basis identification procedure since its invocation.

MSK_DINF_BI_DUAL_TIME
Time spent within the dual phase basis identification procedure since its invocation.

MSK_DINF_BI_CLEAN_TIME
Time spent within the clean-up phase of the basis identification procedure since its invocation.

MSK_DINF_BI_CLEAN_PRIMAL_TIME
Time spent within the primal clean-up optimizer of the basis identification procedure since its invocation.

MSK_DINF_BI_CLEAN_PRIMAL_DUAL_TIME
Time spent within the primal-dual clean-up optimizer of the basis identification procedure since its invocation.

MSK_DINF_BI_CLEAN_DUAL_TIME
Time spent within the dual clean-up optimizer of the basis identification procedure since its invocation.

MSK_DINF_INTPNT_TIME
Time spent within the interior-point optimizer since its invocation.

MSK_DINF_INTPNT_ORDER_TIME
Order time (in seconds).

MSK_DINF_INTPNT_PRIMAL_OBJ
Primal objective value reported by the interior-point optimizer.

MSK_DINF_INTPNT_DUAL_OBJ
Dual objective value reported by the interior-point optimizer.

MSK_DINF_INTPNT_PRIMAL_FEAS
Primal feasibility measure reported by the interior-point optimizers. (For the interior-point optimizer this measure does not directly related to the original problem because a homogeneous model is employed).

MSK_DINF_INTPNT_DUAL_FEAS
Dual feasibility measure reported by the interior-point optimizer. (For the interior-point optimizer this measure does not directly related to the original problem because a homogeneous model is employed.)

MSK_DINF_INTPNT_OPT_STATUS
This measure should converge to $+1$ if the problem has a primal-dual optimal solution, and converge to -1 if problem is (strictly) primal or dual infeasible. If the measure converges to another constant, or fails to settle, the problem is usually ill-posed.

MSK_DINF_SIM_TIME
Time spent in the simplex optimizer since invoking it.

MSK_DINF_SIM_PRIMAL_TIME
Time spent in the primal simplex optimizer since invoking it.

MSK_DINF_SIM_DUAL_TIME
Time spent in the dual simplex optimizer since invoking it.

MSK_DINF_SIM_PRIMAL_DUAL_TIME
Time spent in the primal-dual simplex optimizer since invoking it.

MSK_DINF_SIM_OBJ
Objective value reported by the simplex optimizer.

MSK_DINF_SIM_FEAS
Feasibility measure reported by the simplex optimizer.

MSK_DINF_MIO_TIME
Time spent in the mixed-integer optimizer.

MSK_DINF_MIO_ROOT_PRESOLVE_TIME
Time spent in while presolving the root relaxation.

MSK_DINF_MIO_ROOT_OPTIMIZER_TIME
: Time spent in the optimizer while solving the root relaxation.

MSK_DINF_MIO_OPTIMIZER_TIME
: Total time spent in the optimizer.

MSK_DINF_MIO_HEURISTIC_TIME
: Total time spent in the optimizer.

MSK_DINF_TO_CONIC_TIME
: Time spent in the last to conic reformulation.

MSK_DINF_MIO_CONSTRUCT_SOLUTION_OBJ
: If **MOSEK** has successfully constructed an integer feasible solution, then this item contains the optimal objective value corresponding to the feasible solution.

MSK_DINF_MIO_OBJ_INT
: The primal objective value corresponding to the best integer feasible solution. Please note that at least one integer feasible solution must have located i.e. check | *MSK_IINF_MIO_NUM_INT_SOLUTIONS* |.

MSK_DINF_MIO_OBJ_BOUND
: The best known bound on the objective function. This value is undefined until at least one relaxation has been solved: To see if this is the case check that | *MSK_IINF_MIO_NUM_RELAX* | is strictly positive.

MSK_DINF_MIO_OBJ_REL_GAP
: Given that the mixed-integer optimizer has computed a feasible solution and a bound on the optimal objective value, then this item contains the relative gap defined by

$$\frac{|(\text{objective value of feasible solution}) - (\text{objective bound})|}{\max(\delta, |(\text{objective value of feasible solution})|)}.$$

where δ is given by the parameter *MSK_DPAR_MIO_REL_GAP_CONST*. Otherwise it has the value -1.0.

MSK_DINF_MIO_OBJ_ABS_GAP
: Given the mixed-integer optimizer has computed a feasible solution and a bound on the optimal objective value, then this item contains the absolute gap defined by

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

Otherwise it has the value -1.0.

MSK_DINF_MIO_USER_OBJ_CUT
: If the objective cut is used, then this information item has the value of the cut.

MSK_DINF_MIO_CMIR_SEPARATION_TIME
: Seperation time for CMIR cuts.

MSK_DINF_MIO_CLIQUE_SEPARATION_TIME
: Seperation time for clique cuts.

MSK_DINF_MIO_KNAPSACK_COVER_SEPARATION_TIME
: Seperation time for knapsack cover.

MSK_DINF_MIO_GMI_SEPARATION_TIME
: Seperation time for GMI cuts.

MSK_DINF_MIO_IMPLIED_BOUND_TIME
: Seperation time for implied bound cuts.

MSK_DINF_MIO_ROOT_CUTGEN_TIME
: Total time for cut generation.

MSK_DINF_MIO_PROBING_TIME
: Total time for probing.

MSK_DINF_OPTIMIZER_TIME
Total time spent in the optimizer since it was invoked.

MSK_DINF_PRESOLVE_TIME
Total time (in seconds) spent in the presolve since it was invoked.

MSK_DINF_MIO_DUAL_BOUND_AFTER_PRESOLVE
Value of the dual bound after presolve but before cut generation.

MSK_DINF_PRESOLVE_ELI_TIME
Total time spent in the eliminator since the presolve was invoked.

MSK_DINF_PRESOLVE_LINDEP_TIME
Total time spent in the linear dependency checker since the presolve was invoked.

MSK_DINF_RD_TIME
Time spent reading the data file.

MSK_DINF_SOL_ITR_PRIMAL_OBJ
Primal objective value of the interior-point solution.

MSK_DINF_SOL_ITR_PVIOLCON
Maximal primal bound violation for x^c in the interior-point solution.

MSK_DINF_SOL_ITR_PVIOLVAR
Maximal primal bound violation for x^x in the interior-point solution.

MSK_DINF_SOL_ITR_PVIOLBARVAR
Maximal primal bound violation for $\overline{X}$ in the interior-point solution.

MSK_DINF_SOL_ITR_PVIOLCONES
Maximal primal violation for primal conic constraints in the interior-point solution.

MSK_DINF_SOL_ITR_DUAL_OBJ
Dual objective value of the interior-point solution.

MSK_DINF_SOL_ITR_DVIOLCON
Maximal dual bound violation for x^c in the interior-point solution.

MSK_DINF_SOL_ITR_DVIOLVAR
Maximal dual bound violation for x^x in the interior-point solution.

MSK_DINF_SOL_ITR_DVIOLBARVAR
Maximal dual bound violation for $\overline{X}$ in the interior-point solution.

MSK_DINF_SOL_ITR_DVIOLCONES
Maximal dual violation for dual conic constraints in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_XC
Infinity norm of x^c in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_XX
Infinity norm of x^x in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_BARX
Infinity norm of $\overline{X}$ in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_Y
Infinity norm of y in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_SLC
Infinity norm of s_l^c in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_SUC
Infinity norm of s_u^c in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_SLX
Infinity norm of s_l^x in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_SUX
Infinity norm of s_u^X in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_SNX
Infinity norm of s_n^x in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_BARS
Infinity norm of $\overline{S}$ in the interior-point solution.

MSK_DINF_SOL_BAS_PRIMAL_OBJ
Primal objective value of the basic solution.

MSK_DINF_SOL_BAS_PVIOLCON
Maximal primal bound violation for x^c in the basic solution.

MSK_DINF_SOL_BAS_PVIOLVAR
Maximal primal bound violation for x^x in the basic solution.

MSK_DINF_SOL_BAS_DUAL_OBJ
Dual objective value of the basic solution.

MSK_DINF_SOL_BAS_DVIOLCON
Maximal dual bound violation for x^c in the basic solution.

MSK_DINF_SOL_BAS_DVIOLVAR
Maximal dual bound violation for x^x in the basic solution.

MSK_DINF_SOL_BAS_NRM_XC
Infinity norm of x^c in the basic solution.

MSK_DINF_SOL_BAS_NRM_XX
Infinity norm of x^x in the basic solution.

MSK_DINF_SOL_BAS_NRM_BARX
Infinity norm of $\overline{X}$ in the basic solution.

MSK_DINF_SOL_BAS_NRM_Y
Infinity norm of y in the basic solution.

MSK_DINF_SOL_BAS_NRM_SLC
Infinity norm of s_l^c in the basic solution.

MSK_DINF_SOL_BAS_NRM_SUC
Infinity norm of s_u^c in the basic solution.

MSK_DINF_SOL_BAS_NRM_SLX
Infinity norm of s_l^x in the basic solution.

MSK_DINF_SOL_BAS_NRM_SUX
Infinity norm of s_u^X in the basic solution.

MSK_DINF_SOL_ITG_PRIMAL_OBJ
Primal objective value of the integer solution.

MSK_DINF_SOL_ITG_PVIOLCON
Maximal primal bound violation for x^c in the integer solution.

MSK_DINF_SOL_ITG_PVIOLVAR
Maximal primal bound violation for x^x in the integer solution.

MSK_DINF_SOL_ITG_PVIOLBARVAR
Maximal primal bound violation for $\overline{X}$ in the integer solution.

MSK_DINF_SOL_ITG_PVIOLCONES
Maximal primal violation for primal conic constraints in the integer solution.

MSK_DINF_SOL_ITG_PVIOLITG
Maximal violation for the integer constraints in the integer solution.

MSK_DINF_SOL_ITG_NRM_XC
Infinity norm of x^c in the integer solution.

MSK_DINF_SOL_ITG_NRM_XX
Infinity norm of x^x in the integer solution.

MSK_DINF_SOL_ITG_NRM_BARX
Infinity norm of $\overline{X}$ in the integer solution.

MSK_DINF_INTPNT_FACTOR_NUM_FLOPS
An estimate of the number of flops used in the factorization.

MSK_DINF_QCQO_REFORMULATE_TIME
Time spent with conic quadratic reformulation.

MSK_DINF_QCQO_REFORMULATE_MAX_PERTURBATION
Maximum absolute diagonal perturbation occuring during the QCQO reformulation.

MSK_DINF_QCQO_REFORMULATE_WORST_CHOLESKY_DIAG_SCALING
Worst Cholesky diagonal scaling.

MSK_DINF_QCQO_REFORMULATE_WORST_CHOLESKY_COLUMN_SCALING
Worst Cholesky column scaling.

MSK_DINF_PRIMAL_REPAIR_PENALTY_OBJ
The optimal objective value of the penalty function.

MSKfeaturee
License feature

MSK_FEATURE_PTS
Base system.

MSK_FEATURE_PTON
Nonlinear extension.

MSKliinfiteme
Long integer information items.

MSK_LIINF_MIO_PRESOLVED_ANZ
Number of non-zero entries in the constraint matrix of presolved problem.

MSK_LIINF_MIO_SIMPLEX_ITER
Number of simplex iterations performed by the mixed-integer optimizer.

MSK_LIINF_MIO_INTPNT_ITER
Number of interior-point iterations performed by the mixed-integer optimizer.

MSK_LIINF_BI_PRIMAL_ITER
Number of primal pivots performed in the basis identification.

MSK_LIINF_BI_DUAL_ITER
Number of dual pivots performed in the basis identification.

MSK_LIINF_BI_CLEAN_PRIMAL_ITER
Number of primal clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_PRIMAL_DUAL_ITER
Number of primal-dual clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_DUAL_ITER
Number of dual clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_PRIMAL_DEG_ITER
Number of primal degenerate clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_PRIMAL_DUAL_SUB_ITER
Number of primal-dual subproblem clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_PRIMAL_DUAL_DEG_ITER
Number of primal-dual degenerate clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_DUAL_DEG_ITER
Number of dual degenerate clean iterations performed in the basis identification.

MSK_LIINF_INTPNT_FACTOR_NUM_NZ
Number of non-zeros in factorization.

MSK_LIINF_RD_NUMANZ
Number of non-zeros in A that is read.

MSK_LIINF_RD_NUMQNZ
Number of Q non-zeros.

MSK_LIINF_MIO_SIM_MAXITER_SETBACKS
Number of times the the simplex optimizer has hit the maximum iteration limit when re-optimizing.

MSKiinfiteme
Integer information items.

MSK_IINF_ANA_PRO_NUM_CON
Number of constraints in the problem.

MSK_IINF_ANA_PRO_NUM_CON_LO
Number of constraints with a lower bound and an infinite upper bound.

MSK_IINF_ANA_PRO_NUM_CON_UP
Number of constraints with an upper bound and an infinite lower bound.

MSK_IINF_ANA_PRO_NUM_CON_RA
Number of constraints with finite lower and upper bounds.

MSK_IINF_ANA_PRO_NUM_CON_EQ
Number of equality constraints.

MSK_IINF_ANA_PRO_NUM_CON_FR
Number of unbounded constraints.

MSK_IINF_ANA_PRO_NUM_VAR
Number of variables in the problem.

MSK_IINF_ANA_PRO_NUM_VAR_LO
Number of variables with a lower bound and an infinite upper bound.

MSK_IINF_ANA_PRO_NUM_VAR_UP
Number of variables with an upper bound and an infinite lower bound. This value is set by

MSK_IINF_ANA_PRO_NUM_VAR_RA
Number of variables with finite lower and upper bounds.

MSK_IINF_ANA_PRO_NUM_VAR_EQ
Number of fixed variables.

MSK_IINF_ANA_PRO_NUM_VAR_FR
Number of free variables.

MSK_IINF_ANA_PRO_NUM_VAR_CONT
Number of continuous variables.

MSK_IINF_ANA_PRO_NUM_VAR_BIN
Number of binary (0-1) variables.

MSK_IINF_ANA_PRO_NUM_VAR_INT
Number of general integer variables.

MSK_IINF_OPTIMIZE_RESPONSE
The response code returned by optimize.

MSK_IINF_INTPNT_ITER
Number of interior-point iterations since invoking the interior-point optimizer.

MSK_IINF_INTPNT_FACTOR_DIM_DENSE
Dimension of the dense sub system in factorization.

MSK_IINF_INTPNT_SOLVE_DUAL
Non-zero if the interior-point optimizer is solving the dual problem.

MSK_IINF_MIO_NODE_DEPTH
Depth of the last node solved.

MSK_IINF_MIO_NUMCON
Number of constraints in the problem solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMVAR
Number of variables in the problem solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMINT
Number of integer variables in the problem solved be the mixed-integer optimizer.

MSK_IINF_MIO_PRESOLVED_NUMCONT
Number of continuous variables in the problem solved be the mixed-integer optimizer.

MSK_IINF_MIO_PRESOLVED_NUMBIN
Number of binary variables in the problem solved be the mixed-integer optimizer.

MSK_IINF_MIO_PRESOLVED_NUMCON
Number of constraints in the presolved problem.

MSK_IINF_MIO_PRESOLVED_NUMVAR
Number of variables in the presolved problem.

MSK_IINF_MIO_PRESOLVED_NUMINT
Number of integer variables in the presolved problem.

MSK_IINF_MIO_CLIQUE_TABLE_SIZE
Size of the clique table.

MSK_IINF_MIO_CONSTRUCT_SOLUTION
If this item has the value 0, then **MOSEK** did not try to construct an initial integer feasible solution. If the item has a positive value, then **MOSEK** successfully constructed an initial integer feasible solution.

MSK_IINF_MIO_CONSTRUCT_NUM_ROUNDINGS
Number of values in the integer solution that is rounded to an integer value.

MSK_IINF_MIO_NUM_INT_SOLUTIONS
Number of integer feasible solutions that has been found.

MSK_IINF_MIO_OBJ_BOUND_DEFINED
Non-zero if a valid objective bound has been found, otherwise zero.

MSK_IINF_MIO_NUM_ACTIVE_NODES
Number of active branch bound nodes.

MSK_IINF_MIO_NUM_RELAX
Number of relaxations solved during the optimization.

MSK_IINF_MIO_NUM_BRANCH
Number of branches performed during the optimization.

MSK_IINF_MIO_TOTAL_NUM_CUTS
Total number of cuts generated by the mixed-integer optimizer.

MSK_IINF_MIO_NUM_CMIR_CUTS
Number of Complemented Mixed Integer Rounding (CMIR) cuts.

MSK_IINF_MIO_NUM_CLIQUE_CUTS
Number of clique cuts.

MSK_IINF_MIO_NUM_IMPLIED_BOUND_CUTS
Number of implied bound cuts.

MSK_IINF_MIO_NUM_KNAPSACK_COVER_CUTS
Number of clique cuts.

MSK_IINF_MIO_NUM_GOMORY_CUTS
Number of Gomory cuts.

MSK_IINF_MIO_NUM_REPEATED_PRESOLVE
Number of times presolve was repeated at root.

MSK_IINF_MIO_INITIAL_SOLUTION
Is non-zero if an initial integer solution is specified.

MSK_IINF_MIO_USER_OBJ_CUT
If it is non-zero, then the objective cut is used.

MSK_IINF_MIO_RELGAP_SATISFIED
Non-zero if relative gap is within tolerances.

MSK_IINF_MIO_ABSGAP_SATISFIED
Non-zero if absolute gap is within tolerances.

MSK_IINF_MIO_NEAR_RELGAP_SATISFIED
Non-zero if relative gap is within relaxed tolerances.

MSK_IINF_MIO_NEAR_ABSGAP_SATISFIED
Non-zero if absolute gap is within relaxed tolerances.

MSK_IINF_RD_PROTYPE
Problem type.

MSK_IINF_RD_NUMCON
Number of constraints read.

MSK_IINF_RD_NUMVAR
Number of variables read.

MSK_IINF_RD_NUMBARVAR
Number of variables read.

MSK_IINF_RD_NUMINTVAR
Number of integer-constrained variables read.

MSK_IINF_RD_NUMQ
Number of nonempty Q matrices read.

MSK_IINF_SIM_DUAL_DEG_ITER
The number of dual degenerate iterations.

MSK_IINF_SIM_DUAL_INF_ITER
The number of iterations taken with dual infeasibility.

MSK_IINF_SIM_DUAL_HOTSTART_LU
If 1 then a valid basis factorization of full rank was located and used by the dual simplex algorithm.

MSK_IINF_SIM_PRIMAL_ITER
Number of primal simplex iterations during the last optimization.

MSK_IINF_SIM_DUAL_ITER
Number of dual simplex iterations during the last optimization.

MSK_IINF_SIM_PRIMAL_DUAL_ITER
Number of primal dual simplex iterations during the last optimization.

MSK_IINF_INTPNT_NUM_THREADS
Number of threads that the interior-point optimizer is using.

MSK_IINF_SIM_PRIMAL_INF_ITER
The number of iterations taken with primal infeasibility.

MSK_IINF_SIM_PRIMAL_DUAL_INF_ITER
The number of master iterations with dual infeasibility taken by the primal dual simplex algorithm.

MSK_IINF_SIM_PRIMAL_DEG_ITER
The number of primal degenerate iterations.

MSK_IINF_SIM_PRIMAL_DUAL_DEG_ITER
The number of degenerate major iterations taken by the primal dual simplex algorithm.

MSK_IINF_SIM_PRIMAL_HOTSTART
If 1 then the primal simplex algorithm is solving from an advanced basis.

MSK_IINF_SIM_PRIMAL_HOTSTART_LU
If 1 then a valid basis factorization of full rank was located and used by the primal simplex algorithm.

MSK_IINF_SIM_DUAL_HOTSTART
If 1 then the dual simplex algorithm is solving from an advanced basis.

MSK_IINF_SIM_PRIMAL_DUAL_HOTSTART
If 1 then the primal dual simplex algorithm is solving from an advanced basis.

MSK_IINF_SIM_PRIMAL_DUAL_HOTSTART_LU
If 1 then a valid basis factorization of full rank was located and used by the primal dual simplex algorithm.

MSK_IINF_SOL_ITR_PROSTA
Problem status of the interior-point solution. Updated after each optimization.

MSK_IINF_SOL_ITR_SOLSTA
Solution status of the interior-point solution. Updated after each optimization.

MSK_IINF_SOL_BAS_PROSTA
Problem status of the basic solution. Updated after each optimization.

MSK_IINF_SOL_BAS_SOLSTA
Solution status of the basic solution. Updated after each optimization.

MSK_IINF_SOL_ITG_PROSTA
Problem status of the integer solution. Updated after each optimization.

MSK_IINF_SOL_ITG_SOLSTA
Solution status of the integer solution. Updated after each optimization.

MSK_IINF_SIM_NUMCON
Number of constraints in the problem solved by the simplex optimizer.

MSK_IINF_SIM_NUMVAR
Number of variables in the problem solved by the simplex optimizer.

MSK_IINF_OPT_NUMCON
Number of constraints in the problem solved when the optimizer is called.

MSK_IINF_OPT_NUMVAR
Number of variables in the problem solved when the optimizer is called

MSK_IINF_STO_NUM_A_REALLOC
Number of times the storage for storing A has been changed. A large value may indicates that memory fragmentation may occur.

MSK_IINF_RD_NUMCONE
Number of conic constraints read.

MSK_IINF_SIM_SOLVE_DUAL
Is non-zero if dual problem is solved.

MSKinftypee
Information item types

MSK_INF_DOU_TYPE
Is a double information type.

MSK_INF_INT_TYPE
Is an integer.

MSK_INF_LINT_TYPE
Is a long integer.

MSKiomodee
Input/output modes

MSK_IOMODE_READ
The file is read-only.

MSK_IOMODE_WRITE
The file is write-only. If the file exists then it is truncated when it is opened. Otherwise it is created when it is opened.

MSK_IOMODE_READWRITE
The file is to read and written.

MSKbranchdire
Specifies the branching direction.

MSK_BRANCH_DIR_FREE
The mixed-integer optimizer decides which branch to choose.

MSK_BRANCH_DIR_UP
The mixed-integer optimizer always chooses the up branch first.

MSK_BRANCH_DIR_DOWN
The mixed-integer optimizer always chooses the down branch first.

MSK_BRANCH_DIR_NEAR
Branch in direction nearest to selected fractional variable.

MSK_BRANCH_DIR_FAR
Branch in direction farthest from selected fractional variable.

MSK_BRANCH_DIR_ROOT_LP
Chose direction based on root lp value of selected variable.

MSK_BRANCH_DIR_GUIDED
Branch in direction of current incumbent.

MSK_BRANCH_DIR_PSEUDOCOST
Branch based on the pseudocost of the variable.

MSKmiocontsoltypee
Continuous mixed-integer solution type

MSK_MIO_CONT_SOL_NONE
No interior-point or basic solution are reported when the mixed-integer optimizer is used.

MSK_MIO_CONT_SOL_ROOT
The reported interior-point and basic solutions are a solution to the root node problem when mixed-integer optimizer is used.

MSK_MIO_CONT_SOL_ITG
The reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. A solution is only reported in case the problem has a primal feasible solution.

MSK_MIO_CONT_SOL_ITG_REL
In case the problem is primal feasible then the reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. If the problem is primal infeasible, then the solution to the root node problem is reported.

MSKmiomodee
Integer restrictions

MSK_MIO_MODE_IGNORED
The integer constraints are ignored and the problem is solved as a continuous problem.

MSK_MIO_MODE_SATISFIED
Integer restrictions should be satisfied.

MSKmionodeseltypee
Mixed-integer node selection types

MSK_MIO_NODE_SELECTION_FREE
The optimizer decides the node selection strategy.

MSK_MIO_NODE_SELECTION_FIRST
The optimizer employs a depth first node selection strategy.

MSK_MIO_NODE_SELECTION_BEST
The optimizer employs a best bound node selection strategy.

MSK_MIO_NODE_SELECTION_WORST
The optimizer employs a worst bound node selection strategy.

MSK_MIO_NODE_SELECTION_HYBRID
The optimizer employs a hybrid strategy.

MSK_MIO_NODE_SELECTION_PSEUDO
The optimizer employs selects the node based on a pseudo cost estimate.

MSKmpsformate
MPS file format type

MSK_MPS_FORMAT_STRICT
It is assumed that the input file satisfies the MPS format strictly.

MSK_MPS_FORMAT_RELAXED
It is assumed that the input file satisfies a slightly relaxed version of the MPS format.

MSK_MPS_FORMAT_FREE
It is assumed that the input file satisfies the free MPS format. This implies that spaces are not allowed in names. Otherwise the format is free.

MSK_MPS_FORMAT_CPLEX
The CPLEX compatible version of the MPS format is employed.

MSKmsgkeye
Message keys

MSK_MSG_READING_FILE

MSK_MSG_WRITING_FILE

MSK_MSG_MPS_SELECTED

MSKobjsensee
Objective sense types

MSK_OBJECTIVE_SENSE_MINIMIZE
The problem should be minimized.

MSK_OBJECTIVE_SENSE_MAXIMIZE
The problem should be maximized.

MSKonoffkeye
On/off

MSK_ON
Switch the option on.

MSK_OFF
Switch the option off.

MSKoptimizertypee
Optimizer types

MSK_OPTIMIZER_FREE
The optimizer is chosen automatically.

MSK_OPTIMIZER_INTPNT
The interior-point optimizer is used.

MSK_OPTIMIZER_CONIC
The optimizer for problems having conic constraints.

MSK_OPTIMIZER_PRIMAL_SIMPLEX
The primal simplex optimizer is used.

MSK_OPTIMIZER_DUAL_SIMPLEX
The dual simplex optimizer is used.

MSK_OPTIMIZER_FREE_SIMPLEX
One of the simplex optimizers is used.

MSK_OPTIMIZER_MIXED_INT
The mixed-integer optimizer.

MSKorderingtypee
Ordering strategies

MSK_ORDER_METHOD_FREE
The ordering method is chosen automatically.

MSK_ORDER_METHOD_APPMINLOC
Approximate minimum local fill-in ordering is employed.

MSK_ORDER_METHOD_EXPERIMENTAL
This option should not be used.

MSK_ORDER_METHOD_TRY_GRAPHPAR
Always try the graph partitioning based ordering.

MSK_ORDER_METHOD_FORCE_GRAPHPAR
Always use the graph partitioning based ordering even if it is worse than the approximate minimum local fill ordering.

MSK_ORDER_METHOD_NONE
No ordering is used.

MSKpresolvemodee
Presolve method.

MSK_PRESOLVE_MODE_OFF
The problem is not presolved before it is optimized.

MSK_PRESOLVE_MODE_ON
The problem is presolved before it is optimized.

MSK_PRESOLVE_MODE_FREE
It is decided automatically whether to presolve before the problem is optimized.

MSKparametertypee
Parameter type

MSK_PAR_INVALID_TYPE
Not a valid parameter.

MSK_PAR_DOU_TYPE
Is a double parameter.

MSK_PAR_INT_TYPE
Is an integer parameter.

MSK_PAR_STR_TYPE
Is a string parameter.

MSKproblemiteme
Problem data items

MSK_PI_VAR
Item is a variable.

MSK_PI_CON
Item is a constraint.

MSK_PI_CONE
Item is a cone.

MSKproblemtypee
Problem types

MSK_PROBTYPE_LO
The problem is a linear optimization problem.

MSK_PROBTYPE_QO
The problem is a quadratic optimization problem.

MSK_PROBTYPE_QCQO
The problem is a quadratically constrained optimization problem.

MSK_PROBTYPE_GECO
General convex optimization.

MSK_PROBTYPE_CONIC
A conic optimization.

MSK_PROBTYPE_MIXED
General nonlinear constraints and conic constraints. This combination can not be solved by **MOSEK**.

MSKprostae
Problem status keys

MSK_PRO_STA_UNKNOWN
Unknown problem status.

MSK_PRO_STA_PRIM_AND_DUAL_FEAS
The problem is primal and dual feasible.

MSK_PRO_STA_PRIM_FEAS
The problem is primal feasible.

MSK_PRO_STA_DUAL_FEAS
The problem is dual feasible.

MSK_PRO_STA_NEAR_PRIM_AND_DUAL_FEAS
The problem is at least nearly primal and dual feasible.

MSK_PRO_STA_NEAR_PRIM_FEAS
The problem is at least nearly primal feasible.

MSK_PRO_STA_NEAR_DUAL_FEAS
The problem is at least nearly dual feasible.

MSK_PRO_STA_PRIM_INFEAS
The problem is primal infeasible.

MSK_PRO_STA_DUAL_INFEAS
The problem is dual infeasible.

MSK_PRO_STA_PRIM_AND_DUAL_INFEAS
The problem is primal and dual infeasible.

MSK_PRO_STA_ILL_POSED
The problem is ill-posed. For example, it may be primal and dual feasible but have a positive duality gap.

MSK_PRO_STA_PRIM_INFEAS_OR_UNBOUNDED
The problem is either primal infeasible or unbounded. This may occur for mixed-integer problems.

MSKxmlwriteroutputtypee
XML writer output mode

MSK_WRITE_XML_MODE_ROW
Write in row order.

MSK_WRITE_XML_MODE_COL
Write in column order.

MSKrescodetypee
Response code type

MSK_RESPONSE_OK
The response code is OK.

MSK_RESPONSE_WRN
The response code is a warning.

MSK_RESPONSE_TRM
The response code is an optimizer termination status.

MSK_RESPONSE_ERR
The response code is an error.

MSK_RESPONSE_UNK
The response code does not belong to any class.

MSKscalingtypee
Scaling type

MSK_SCALING_FREE
The optimizer chooses the scaling heuristic.

MSK_SCALING_NONE
No scaling is performed.

MSK_SCALING_MODERATE
A conservative scaling is performed.

MSK_SCALING_AGGRESSIVE
A very aggressive scaling is performed.

MSKscalingmethode
Scaling method

MSK_SCALING_METHOD_POW2
Scales only with power of 2 leaving the mantissa untouched.

MSK_SCALING_METHOD_FREE
The optimizer chooses the scaling heuristic.

MSKsensitivitytypee
Sensitivity types

MSK_SENSITIVITY_TYPE_BASIS
Basis sensitivity analysis is performed.

MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION
Optimal partition sensitivity analysis is performed.

MSKsimseltypee
Simplex selection strategy

MSK_SIM_SELECTION_FREE
The optimizer chooses the pricing strategy.

MSK_SIM_SELECTION_FULL
The optimizer uses full pricing.

MSK_SIM_SELECTION_ASE
The optimizer uses approximate steepest-edge pricing.

MSK_SIM_SELECTION_DEVEX
The optimizer uses devex steepest-edge pricing (or if it is not available an approximate steep-edge selection).

MSK_SIM_SELECTION_SE
The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

MSK_SIM_SELECTION_PARTIAL
The optimizer uses a partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.

MSKsoliteme
Solution items

MSK_SOL_ITEM_XC
Solution for the constraints.

MSK_SOL_ITEM_XX
Variable solution.

MSK_SOL_ITEM_Y
Lagrange multipliers for equations.

MSK_SOL_ITEM_SLC
Lagrange multipliers for lower bounds on the constraints.

MSK_SOL_ITEM_SUC
Lagrange multipliers for upper bounds on the constraints.

MSK_SOL_ITEM_SLX
Lagrange multipliers for lower bounds on the variables.

MSK_SOL_ITEM_SUX
Lagrange multipliers for upper bounds on the variables.

MSK_SOL_ITEM_SNX
Lagrange multipliers corresponding to the conic constraints on the variables.

MSKsolstae
Solution status keys

MSK_SOL_STA_UNKNOWN
Status of the solution is unknown.

MSK_SOL_STA_OPTIMAL
The solution is optimal.

MSK_SOL_STA_PRIM_FEAS
The solution is primal feasible.

MSK_SOL_STA_DUAL_FEAS
The solution is dual feasible.

MSK_SOL_STA_PRIM_AND_DUAL_FEAS
The solution is both primal and dual feasible.

MSK_SOL_STA_NEAR_OPTIMAL
The solution is nearly optimal.

MSK_SOL_STA_NEAR_PRIM_FEAS
The solution is nearly primal feasible.

MSK_SOL_STA_NEAR_DUAL_FEAS
The solution is nearly dual feasible.

MSK_SOL_STA_NEAR_PRIM_AND_DUAL_FEAS
The solution is nearly both primal and dual feasible.

MSK_SOL_STA_PRIM_INFEAS_CER
The solution is a certificate of primal infeasibility.

MSK_SOL_STA_DUAL_INFEAS_CER
The solution is a certificate of dual infeasibility.

MSK_SOL_STA_NEAR_PRIM_INFEAS_CER
The solution is almost a certificate of primal infeasibility.

MSK_SOL_STA_NEAR_DUAL_INFEAS_CER
The solution is almost a certificate of dual infeasibility.

MSK_SOL_STA_PRIM_ILLPOSED_CER
The solution is a certificate that the primal problem is illposed.

MSK_SOL_STA_DUAL_ILLPOSED_CER
The solution is a certificate that the dual problem is illposed.

MSK_SOL_STA_INTEGER_OPTIMAL
The primal solution is integer optimal.

MSK_SOL_STA_NEAR_INTEGER_OPTIMAL
The primal solution is near integer optimal.

MSKsoltypee
Solution types

MSK_SOL_BAS
The basic solution.

MSK_SOL_ITR
The interior solution.

MSK_SOL_ITG
The integer solution.

MSKsolveforme
Solve primal or dual form

MSK_SOLVE_FREE
The optimizer is free to solve either the primal or the dual problem.

MSK_SOLVE_PRIMAL
The optimizer should solve the primal problem.

MSK_SOLVE_DUAL
The optimizer should solve the dual problem.

MSKstakeye
Status keys

MSK_SK_UNK
The status for the constraint or variable is unknown.

MSK_SK_BAS
The constraint or variable is in the basis.

MSK_SK_SUPBAS
The constraint or variable is super basic.

MSK_SK_LOW
The constraint or variable is at its lower bound.

MSK_SK_UPR
The constraint or variable is at its upper bound.

MSK_SK_FIX
The constraint or variable is fixed.

MSK_SK_INF
The constraint or variable is infeasible in the bounds.

MSKstartpointtypee
Starting point types

MSK_STARTING_POINT_FREE
The starting point is chosen automatically.

MSK_STARTING_POINT_GUESS
The optimizer guesses a starting point.

MSK_STARTING_POINT_CONSTANT
The optimizer constructs a starting point by assigning a constant value to all primal and dual variables. This starting point is normally robust.

MSK_STARTING_POINT_SATISFY_BOUNDS
The starting point is chosen to satisfy all the simple bounds on nonlinear variables. If this starting point is employed, then more care than usual should employed when choosing the bounds on the nonlinear variables. In particular very tight bounds should be avoided.

MSKstreamtypee
Stream types

MSK_STREAM_LOG
Log stream. Contains the aggregated contents of all other streams. This means that a message written to any other stream will also be written to this stream.

MSK_STREAM_MSG
Message stream. Log information relating to performance and progress of the optimization is written to this stream.

MSK_STREAM_ERR
Error stream. Error messages are written to this stream.

MSK_STREAM_WRN
Warning stream. Warning messages are written to this stream.

MSKvaluee
Integer values

MSK_MAX_STR_LEN
Maximum string length allowed in **MOSEK**.

MSK_LICENSE_BUFFER_LENGTH
The length of a license key buffer.

MSKvariabletypee
Variable types

MSK_VAR_TYPE_CONT
Is a continuous variable.

MSK_VAR_TYPE_INT
Is an integer variable.

CHAPTER

ELEVEN

SUPPORTED FILE FORMATS

MOSEK supports a range of problem and solution formats listed in Table 11.1 and Table 11.2. The **Task format** is **MOSEK**'s native binary format and it supports all features that **MOSEK** supports. The **OPF format** is **MOSEK**'s human-readable alternative that supports nearly all features (everything except semidefinite problems). In general, text formats are significantly slower to read, but can be examined and edited directly in any text editor.

Problem formats

See Table 11.1.

Table 11.1: List of supported file formats for optimization problems.

Format Type	Ext.	Binary/Text	LP	QP	CQO	SDP
LP	lp	plain text	X	X		
MPS	mps	plain text	X	X		
OPF	opf	plain text	X	X	X	
CBF	cbf	plain text	X		X	X
OSiL	xml	xml text	X	X		
Task format	task	binary	X	X	X	X
Jtask format	jtask	text	X	X	X	X

Solution formats

See Table 11.2.

Table 11.2: List of supported solution formats.

Format Type	Ext.	Binary/Text	Description
SOL	sol	plain text	Interior Solution
	bas	plain text	Basic Solution
	int	plain text	Integer
Jsol format	jsol	text	Solution

Compression

MOSEK supports GZIP compression of files. Problem files with an additional `.gz` extension are assumed to be compressed when read, and are automatically compressed when written. For example, a file called

```
problem.mps.gz
```

will be considered as a GZIP compressed MPS file.

11.1 The LP File Format

MOSEK supports the LP file format with some extensions. The LP format is not a completely well-defined standard and hence different optimization packages may interpret the same LP file in slightly different ways. **MOSEK** tries to emulate as closely as possible CPLEX's behavior, but tries to stay backward compatible.

The LP file format can specify problems on the form

$$
\begin{array}{lccccc}
\text{minimize/maximize} & & & c^T x + \frac{1}{2} q^o(x) & & \\
\text{subject to} & l^c & \leq & Ax + \frac{1}{2} q(x) & \leq & u^c, \\
 & l^x & \leq & x & \leq & u^x, \\
 & & & x_{\mathcal{J}} \text{ integer}, & &
\end{array}
$$

where

- $x \in \mathbb{R}^n$ is the vector of decision variables.
- $c \in \mathbb{R}^n$ is the linear term in the objective.
- $q^o :\in \mathbb{R}^n \rightarrow \mathbb{R}$ is the quadratic term in the objective where

$$q^o(x) = x^T Q^o x$$

and it is assumed that

$$Q^o = (Q^o)^T.$$

- $A \in \mathbb{R}^{m \times n}$ is the constraint matrix.
- $l^c \in \mathbb{R}^m$ is the lower limit on the activity for the constraints.
- $u^c \in \mathbb{R}^m$ is the upper limit on the activity for the constraints.
- $l^x \in \mathbb{R}^n$ is the lower limit on the activity for the variables.
- $u^x \in \mathbb{R}^n$ is the upper limit on the activity for the variables.
- $q : \mathbb{R}^n \rightarrow \mathbb{R}$ is a vector of quadratic functions. Hence,

$$q_i(x) = x^T Q^i x$$

where it is assumed that

$$Q^i = (Q^i)^T.$$

- $\mathcal{J} \subseteq \{1, 2, \ldots, n\}$ is an index set of the integer constrained variables.

11.1.1 File Sections

An LP formatted file contains a number of sections specifying the objective, constraints, variable bounds, and variable types. The section keywords may be any mix of upper and lower case letters.

Objective Function

The first section beginning with one of the keywords

```
max
maximum
maximize
min
minimum
minimize
```

defines the objective sense and the objective function, i.e.

$$c^T x + \frac{1}{2} x^T Q^o x.$$

The objective may be given a name by writing

```
myname:
```

before the expressions. If no name is given, then the objective is named `obj`.

The objective function contains linear and quadratic terms. The linear terms are written as:

```
4 x1 + x2 - 0.1 x3
```

and so forth. The quadratic terms are written in square brackets (`[ ]`) and are either squared or multiplied as in the examples

```
x1^2
```

and

```
x1 * x2
```

There may be zero or more pairs of brackets containing quadratic expressions.

An example of an objective section is

```
minimize
myobj: 4 x1 + x2 - 0.1 x3 + [ x1^2 + 2.1 x1 * x2 ]/2
```

Please note that the quadratic expressions are multiplied with $\frac{1}{2}$, so that the above expression means

$$\text{minimize} \quad 4x_1 + x_2 - 0.1 \cdot x_3 + \tfrac{1}{2}(x_1^2 + 2.1 \cdot x_1 \cdot x_2)$$

If the same variable occurs more than once in the linear part, the coefficients are added, so that `4 x1 + 2 x1` is equivalent to `6 x1`. In the quadratic expressions `x1 * x2` is equivalent to `x2 * x1` and, as in the linear part, if the same variables multiplied or squared occur several times their coefficients are added.

Constraints

The second section beginning with one of the keywords

```
subj to
subject to
s.t.
st
```

defines the linear constraint matrix A and the quadratic matrices Q^i.

A constraint contains a name (optional), expressions adhering to the same rules as in the objective and a bound:

```
subject to
con1: x1 + x2 + [ x3^2 ]/2 <= 5.1
```

The bound type (here `<=`) may be any of `<`, `<=`, `=`, `>`, `>=` (`<` and `<=` mean the same), and the bound may be any number.

In the standard LP format it is not possible to define more than one bound, but **MOSEK** supports defining ranged constraints by using double-colon (`::`) instead of a single-colon (`:`) after the constraint name, i.e.

$$-5 \leq x_1 + x_2 \leq 5 \tag{11.1}$$

may be written as

```
con:: -5 < x_1 + x_2 < 5
```

By default **MOSEK** writes ranged constraints this way.

If the files must adhere to the LP standard, ranged constraints must either be split into upper bounded and lower bounded constraints or be written as an equality with a slack variable. For example the expression (11.1) may be written as

$$x_1 + x_2 - sl_1 = 0, \ -5 \leq sl_1 \leq 5.$$

Bounds

Bounds on the variables can be specified in the bound section beginning with one of the keywords

```
bound
bounds
```

The bounds section is optional but should, if present, follow the `subject to` section. All variables listed in the bounds section must occur in either the objective or a constraint.

The default lower and upper bounds are 0 and $+\infty$. A variable may be declared free with the keyword `free`, which means that the lower bound is $-\infty$ and the upper bound is $+\infty$. Furthermore it may be assigned a finite lower and upper bound. The bound definitions for a given variable may be written in one or two lines, and bounds can be any number or $\pm\infty$ (written as `+inf`/`-inf`/`+infinity`/`-infinity`) as in the example

```
bounds
x1 free
x2 <= 5
0.1 <= x2
x3 = 42
2 <= x4 < +inf
```

Variable Types

The final two sections are optional and must begin with one of the keywords

```
bin
binaries
binary
```

and

```
gen
general
```

Under `general` all integer variables are listed, and under `binary` all binary (integer variables with bounds 0 and 1) are listed:

```
general
x1 x2
binary
x3 x4
```

Again, all variables listed in the binary or general sections must occur in either the objective or a constraint.

Terminating Section

Finally, an LP formatted file must be terminated with the keyword

```
end
```

11.1.2 LP File Examples

Linear example `lo1.lp`

```
\ File: lo1.lp
maximize
obj: 3 x1 + x2 + 5 x3 + x4
subject to
c1:  3 x1 + x2 + 2 x3 = 30
c2:  2 x1 + x2 + 3 x3 + x4 >= 15
c3:  2 x2 + 3 x4 <= 25
bounds
 0 <= x1 <= +infinity
 0 <= x2 <= 10
 0 <= x3 <= +infinity
 0 <= x4 <= +infinity
end
```

Mixed integer example `milo1.lp`

```
maximize
obj: x1 + 6.4e-01 x2
subject to
c1:  5e+01 x1 + 3.1e+01 x2 <= 2.5e+02
c2:  3e+00 x1 - 2e+00 x2 >= -4e+00
bounds
 0 <= x1 <= +infinity
 0 <= x2 <= +infinity
general
 x1 x2
end
```

11.1.3 LP Format peculiarities

Comments

Anything on a line after a \ is ignored and is treated as a comment.

Names

A name for an objective, a constraint or a variable may contain the letters *a-z*, *A-Z*, the digits *0-9* and the characters

```
!"#$%&()/,.;?@_'`|~
```

The first character in a name must not be a number, a period or the letter `e` or `E`. Keywords must not be used as names.

MOSEK accepts any character as valid for names, except `\0`. A name that is not allowed in LP file will be changed and a warning will be issued.

The algorithm for making names LP valid works as follows: The name is interpreted as an `utf-8` string. For a unicode character `c`:

- If `c==_` (underscore), the output is `__` (two underscores).
- If `c` is a valid LP name character, the output is just `c`.
- If `c` is another character in the ASCII range, the output is `_XX`, where `XX` is the hexadecimal code for the character.
- If `c` is a character in the range *127-65535*, the output is `_uXXXX`, where `XXXX` is the hexadecimal code for the character.
- If `c` is a character above 65535, the output is `_UXXXXXXXX`, where `XXXXXXXX` is the hexadecimal code for the character.

Invalid `utf-8` substrings are escaped as `_XX'`, and if a name starts with a period, *e* or *E*, that character is escaped as `_XX`.

Variable Bounds

Specifying several upper or lower bounds on one variable is possible but **MOSEK** uses only the tightest bounds. If a variable is fixed (with `=`), then it is considered the tightest bound.

MOSEK Extensions to the LP Format

Some optimization software packages employ a more strict definition of the LP format than the one used by **MOSEK**. The limitations imposed by the strict LP format are the following:

- Quadratic terms in the constraints are not allowed.
- Names can be only 16 characters long.
- Lines must not exceed 255 characters in length.

If an LP formatted file created by **MOSEK** should satisfy the strict definition, then the parameter

- *MSK_IPAR_WRITE_LP_STRICT_FORMAT*

should be set; note, however, that some problems cannot be written correctly as a strict LP formatted file. For instance, all names are truncated to 16 characters and hence they may loose their uniqueness and change the problem.

To get around some of the inconveniences converting from other problem formats, **MOSEK** allows lines to contain 1024 characters and names may have any length (shorter than the 1024 characters).

Internally in **MOSEK** names may contain any (printable) character, many of which cannot be used in LP names. Setting the parameters

- *MSK_IPAR_READ_LP_QUOTED_NAMES* and
- *MSK_IPAR_WRITE_LP_QUOTED_NAMES*

allows **MOSEK** to use quoted names. The first parameter tells **MOSEK** to remove quotes from quoted names e.g, `"x1"`, when reading LP formatted files. The second parameter tells **MOSEK** to put quotes around any semi-illegal name (names beginning with a number or a period) and fully illegal name (containing illegal characters). As double quote is a legal character in the LP format, quoting semi-illegal names makes them legal in the pure LP format as long as they are still shorter than 16 characters. Fully illegal names are still illegal in a pure LP file.

11.1.4 The strict LP format

The LP format is not a formal standard and different vendors have slightly different interpretations of the LP format. To make **MOSEK**'s definition of the LP format more compatible with the definitions of other vendors, use the parameter setting

- *MSK_IPAR_WRITE_LP_STRICT_FORMAT* = *MSK_ON*

This setting may lead to truncation of some names and hence to an invalid LP file. The simple solution to this problem is to use the parameter setting

- *MSK_IPAR_WRITE_GENERIC_NAMES* = *MSK_ON*

which will cause all names to be renamed systematically in the output file.

11.1.5 Formatting of an LP File

A few parameters control the visual formatting of LP files written by **MOSEK** in order to make it easier to read the files. These parameters are

- *MSK_IPAR_WRITE_LP_LINE_WIDTH*
- *MSK_IPAR_WRITE_LP_TERMS_PER_LINE*

The first parameter sets the maximum number of characters on a single line. The default value is 80 corresponding roughly to the width of a standard text document.

The second parameter sets the maximum number of terms per line; a term means a sign, a coefficient, and a name (for example `+ 42 elephants`). The default value is 0, meaning that there is no maximum.

Unnamed Constraints

Reading and writing an LP file with **MOSEK** may change it superficially. If an LP file contains unnamed constraints or objective these are given their generic names when the file is read (however unnamed constraints in **MOSEK** are written without names).

11.2 The MPS File Format

MOSEK supports the standard MPS format with some extensions. For a detailed description of the MPS format see the book by Nazareth *[Naz87]*.

11.2.1 MPS File Structure

The version of the MPS format supported by **MOSEK** allows specification of an optimization problem of the form

$$
\begin{array}{rcccl}
l^c & \leq & Ax + q(x) & \leq & u^c, \\
l^x & \leq & x & \leq & u^x, \\
 & & x \in \mathcal{K}, & & \\
 & & x_{\mathcal{J}} \quad \text{integer}, & &
\end{array}
\tag{11.2}
$$

where

- $x \in \mathbb{R}^n$ is the vector of decision variables.
- $A \in \mathbb{R}^{m \times n}$ is the constraint matrix.
- $l^c \in \mathbb{R}^m$ is the lower limit on the activity for the constraints.
- $u^c \in \mathbb{R}^m$ is the upper limit on the activity for the constraints.
- $l^x \in \mathbb{R}^n$ is the lower limit on the activity for the variables.
- $u^x \in \mathbb{R}^n$ is the upper limit on the activity for the variables.
- $q : \mathbb{R}^n \to \mathbb{R}$ is a vector of quadratic functions. Hence,

$$q_i(x) = \frac{1}{2} x^T Q^i x$$

where it is assumed that

$$Q^i = (Q^i)^T.$$

Please note the explicit $\frac{1}{2}$ in the quadratic term and that Q^i is required to be symmetric.

- $\mathcal{K}$ is a convex cone.
- $\mathcal{J} \subseteq \{1, 2, \ldots, n\}$ is an index set of the integer-constrained variables.

An MPS file with one row and one column can be illustrated like this:

```
*        1         2         3         4         5         6
*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]
QMATRIX
[vname1]  [vname2]    [value1]
QUADOBJ
[vname1]  [vname2]    [value1]
QCMATRIX      [cname1]
[vname1]  [vname2]    [value1]
BOUNDS
?? [name]    [vname1]    [value1]
CSECTION      [kname1]    [value1]     [ktype]
[vname1]
ENDATA
```

Here the names in capitals are keywords of the MPS format and names in brackets are custom defined names or values. A couple of notes on the structure:

- Fields: All items surrounded by brackets appear in *fields*. The fields named "`valueN`" are numerical values. Hence, they must have the format

```
  [+|-]XXXXXXX.XXXXXX[[e|E][+|-]XXX]


where
```

```
.. code-block:: text

   X = [0|1|2|3|4|5|6|7|8|9].
```

- Sections: The MPS file consists of several sections where the names in capitals indicate the beginning of a new section. For example, `COLUMNS` denotes the beginning of the columns section.
- Comments: Lines starting with an `*` are comment lines and are ignored by **MOSEK**.
- Keys: The question marks represent keys to be specified later.
- Extensions: The sections `QSECTION` and `CSECTION` are specific **MOSEK** extensions of the MPS format. The sections `QMATRIX`, `QUADOBJ` and `QCMATRIX` are included for sake of compatibility with other vendors extensions to the MPS format.

The standard MPS format is a fixed format, i.e. everything in the MPS file must be within certain fixed positions. **MOSEK** also supports a *free format*. See Section *11.2.9* for details.

Linear example `lo1.mps`

A concrete example of a MPS file is presented below:

```
* File: lo1.mps
NAME          lo1
OBJSENSE
    MAX
ROWS
 N  obj
 E  c1
 G  c2
 L  c3
COLUMNS
    x1        obj       3
    x1        c1        3
    x1        c2        2
    x2        obj       1
    x2        c1        1
    x2        c2        1
    x2        c3        2
    x3        obj       5
    x3        c1        2
    x3        c2        3
    x4        obj       1
    x4        c2        1
    x4        c3        3
RHS
    rhs       c1        30
    rhs       c2        15
    rhs       c3        25
RANGES
BOUNDS
 UP bound     x2        10
ENDATA
```

Subsequently each individual section in the MPS format is discussed.

Section `NAME`

In this section a name (`[name]`) is assigned to the problem.

OBJSENSE (optional)

This is an optional section that can be used to specify the sense of the objective function. The OBJSENSE section contains one line at most which can be one of the following

```
MIN
MINIMIZE
MAX
MAXIMIZE
```

It should be obvious what the implication is of each of these four lines.

OBJNAME (optional)

This is an optional section that can be used to specify the name of the row that is used as objective function. The OBJNAME section contains one line at most which has the form

```
objname
```

objname should be a valid row name.

ROWS

A record in the ROWS section has the form

```
?  [cname1]
```

where the requirements for the fields are as follows:

Field	Starting Position	Max Width	required	Description
?	2	1	Yes	Constraint key
[cname1]	5	8	Yes	Constraint name

Hence, in this section each constraint is assigned an unique name denoted by [cname1]. Please note that [cname1] starts in position 5 and the field can be at most 8 characters wide. An initial key ? must be present to specify the type of the constraint. The key can have the values E, G, L, or N with the following interpretation:

Constraint type	l_i^c	u_i^c
E	finite	l_i^c
G	finite	∞
L	$-\infty$	finite
N	$-\infty$	∞

In the MPS format an objective vector is not specified explicitly, but one of the constraints having the key N will be used as the objective vector c . In general, if multiple N type constraints are specified, then the first will be used as the objective vector c.

COLUMNS

In this section the elements of A are specified using one or more records having the form:

```
[vname1]  [cname1]    [value1]     [cname2]  [value2]
```

where the requirements for each field are as follows:

Field	Starting Position	Max Width	required	Description
[vname1]	5	8	Yes	Variable name
[cname1]	15	8	Yes	Constraint name
[value1]	25	12	Yes	Numerical value
[cname2]	40	8	No	Constraint name
[value2]	50	12	No	Numerical value

Hence, a record specifies one or two elements a_{ij} of A using the principle that [vname1] and [cname1] determines j and i respectively. Please note that [cname1] must be a constraint name specified in the ROWS section. Finally, [value1] denotes the numerical value of a_{ij} . Another optional element is specified by [cname2], and [value2] for the variable specified by [vname1]. Some important comments are:

- All elements belonging to one variable must be grouped together.
- Zero elements of A should not be specified.
- At least one element for each variable should be specified.

RHS (optional)

A record in this section has the format

```
[name]    [cname1]    [value1]     [cname2]  [value2]
```

where the requirements for each field are as follows:

Field	Starting Position	Max Width	required	Description
[name]	5	8	Yes	Name of the RHS vector
[cname1]	15	8	Yes	Constraint name
[value1]	25	12	Yes	Numerical value
[cname2]	40	8	No	Constraint name
[value2]	50	12	No	Numerical value

The interpretation of a record is that [name] is the name of the RHS vector to be specified. In general, several vectors can be specified. [cname1] denotes a constraint name previously specified in the ROWS section. Now, assume that this name has been assigned to the i th constraint and v_1 denotes the value specified by [value1], then the interpretation of v_1 is:

Constraint type	l_i^c	u_i^c
E	v_1	v_1
G	v_1	
L		v_1
N		

An optional second element is specified by [cname2] and [value2] and is interpreted in the same way. Please note that it is not necessary to specify zero elements, because elements are assumed to be zero.

RANGES (optional)

A record in this section has the form

```
[name]    [cname1]    [value1]     [cname2]  [value2]
```

where the requirements for each fields are as follows:

Field	Starting Position	Max Width	required	Description
[name]	5	8	Yes	Name of the RANGE vector
[cname1]	15	8	Yes	Constraint name
[value1]	25	12	Yes	Numerical value
[cname2]	40	8	No	Constraint name
[value2]	50	12	No	Numerical value

The records in this section are used to modify the bound vectors for the constraints, i.e. the values in l^c and u^c . A record has the following interpretation: `[name]` is the name of the `RANGE` vector and `[cname1]` is a valid constraint name. Assume that `[cname1]` is assigned to the i th constraint and let v_1 be the value specified by `[value1]`, then a record has the interpretation:

Constraint type	Sign of v_1	l_i^c	u_i^c
E	$-$	$u_i^c + v_1$	
E	$+$		$l_i^c + v_1$
G	$-$ or $+$	$l_i^c + \|v_1\|$	
L	$-$ or $+$	$u_i^c - \|v_1\|$	
N			

`QSECTION` (optional)

Within the `QSECTION` the label `[cname1]` must be a constraint name previously specified in the `ROWS` section. The label `[cname1]` denotes the constraint to which the quadratic term belongs. A record in the `QSECTION` has the form

```
[vname1]  [vname2]    [value1]     [vname3]  [value2]
```

where the requirements for each field are:

Field	Starting Position	Max Width	required	Description
`[vname1]`	5	8	Yes	Variable name
`[vname2]`	15	8	Yes	Variable name
`[value1]`	25	12	Yes	Numerical value
`[vname3]`	40	8	No	Variable name
`[value2]`	50	12	No	Numerical value

A record specifies one or two elements in the lower triangular part of the Q^i matrix where `[cname1]` specifies the i . Hence, if the names `[vname1]` and `[vname2]` have been assigned to the k th and j th variable, then Q^i_{kj} is assigned the value given by `[value1]` An optional second element is specified in the same way by the fields `[vname1]`, `[vname3]`, and `[value2]`.

The example

$$\begin{array}{lccl} \text{minimize} & -x_2 + \frac{1}{2}(2x_1^2 - 2x_1x_3 + 0.2x_2^2 + 2x_3^2) & & \\ \text{subject to} & x_1 + x_2 + x_3 & \geq & 1, \\ & x \geq 0 & & \end{array}$$

has the following MPS file representation

```
* File: qo1.mps
NAME          qo1
ROWS
N  obj
G  c1
COLUMNS
x1         c1        1.0
x2         obj       -1.0
x2         c1        1.0
x3         c1        1.0
RHS
rhs        c1        1.0
QSECTION      obj
x1         x1        2.0
x1         x3        -1.0
x2         x2        0.2
x3         x3        2.0
ENDATA
```

Regarding the `QSECTION`s please note that:

- Only one QSECTION is allowed for each constraint.
- The QSECTIONs can appear in an arbitrary order after the COLUMNS section.
- All variable names occurring in the QSECTION must already be specified in the COLUMNS section.
- All entries specified in a QSECTION are assumed to belong to the lower triangular part of the quadratic term of Q .

QMATRIX/QUADOBJ (optional)

The QMATRIX and QUADOBJ sections allow to define the quadratic term of the objective function. They differ in how the quadratic term of the objective function is stored:

- QMATRIX It stores all the nonzeros coefficients, withouot taking advantage of the symmetry of the Q matrix.
- QUADOBJ It only store the upper diagonal nonzero elements of the Q matrix.

A record in both sections has the form:

```
[vname1]  [vname2]    [value1]
```

where the requirements for each field are:

Field	Starting Position	Max Width	required	Description
[vname1]	5	8	Yes	Variable name
[vname2]	15	8	Yes	Variable name
[value1]	25	12	Yes	Numerical value

A record specifies one elements of the Q matrix in the objective function . Hence, if the names [vname1] and [vname2] have been assigned to the k th and j th variable, then Q_{kj} is assigned the value given by [value1]. Note that a line must apper for each off-diagonal coefficient if using a QMATRIX section, while only one entry is required in a QUADOBJ section. The quadratic part of the objective function will be evaluated as $1/2x^TQx$.

The example

$$\begin{array}{lccc} \text{minimize} & -x_2 + \frac{1}{2}(2x_1^2 - 2x_1x_3 + 0.2x_2^2 + 2x_3^2) & & \\ \text{subject to} & x_1 + x_2 + x_3 & \geq & 1, \\ & x \geq 0 & & \end{array}$$

has the following MPS file representation using QMATRIX

```
* File: qo1_matrix.mps
NAME          qo1_qmatrix
ROWS
 N  obj
 G  c1
COLUMNS
    x1        c1        1.0
    x2        obj       -1.0
    x2        c1        1.0
    x3        c1        1.0
RHS
    rhs       c1        1.0
QMATRIX
    x1        x1        2.0
    x1        x3        -1.0
    x3        x1        -1.0
    x2        x2        0.2
    x3        x3        2.0
ENDATA
```

or the following using QUADOBJ

```
* File: qo1_quadobj.mps
NAME          qo1_quadobj
ROWS
 N  obj
 G  c1
COLUMNS
    x1        c1        1.0
    x2        obj       -1.0
    x2        c1        1.0
    x3        c1        1.0
RHS
    rhs       c1        1.0
QUADOBJ
    x1        x1        2.0
    x1        x3        -1.0
    x2        x2        0.2
    x3        x3        2.0
ENDATA
```

Please also note that:

- A QMATRIX/QUADOBJ section can appear in an arbitrary order after the COLUMNS section.
- All variable names occurring in the QMATRIX/QUADOBJ section must already be specified in the COLUMNS section.

11.2.2 QCMATRIX (optional)

A QCMATRIX section allows to specify the quadratic part of a given constraints. Within the QCMATRIX the label [cname1] must be a constraint name previously specified in the ROWS section. The label [cname1] denotes the constraint to which the quadratic term belongs. A record in the QSECTION has the form

```
[vname1]  [vname2]    [value1]
```

where the requirements for each field are:

Field	Starting Position	Max Width	required	Description
[vname1]	5	8	Yes	Variable name
[vname2]	15	8	Yes	Variable name
[value1]	25	12	Yes	Numerical value

A record specifies an entry of the Q^i matrix where [cname1] specifies the i. Hence, if the names [vname1] and [vname2] have been assigned to the k th and j th variable, then Q^i_{kj} is assigned the value given by [value1]. Moreover, the quadratic term is represented as $1/2x^TQx$.

The example

$$
\begin{array}{lccc}
\text{minimize} & x_2 & & \\
\text{subject to} & x_1 + x_2 + x_3 & \geq & 1, \\
 & \frac{1}{2}(-2x_1x_3 + 0.2x_2^2 + 2x_3^2) & \leq & 10, \\
 & x \geq 0 & &
\end{array}
$$

has the following MPS file representation

```
* File: qo1.mps
NAME          qo1
ROWS
 N  obj
 G  c1
 L  q1
COLUMNS
```

```
    x1        c1        1.0
    x2        obj       -1.0
    x2        c1        1.0
    x3        c1        1.0
RHS
    rhs       c1        1.0
    rhs       q1        10.0
QCMATRIX      q1
    x1        x1        2.0
    x1        x3        -1.0
    x3        x1        -1.0
    x2        x2        0.2
    x3        x3        2.0
ENDATA
```

Regarding the QCMATRIXs please note that:

- Only one QCMATRIX is allowed for each constraint.
- The QCMATRIXs can appear in an arbitrary order after the COLUMNS section.
- All variable names occurring in the QSECTION must already be specified in the COLUMNS section.
- A QCMATRIX does not exploit the symmetry of Q: an off-diagonal entry (i, j) should apperas twice.

11.2.3 BOUNDS (optional)

In the BOUNDS section changes to the default bounds vectors l^x and u^x are specified. The default bounds vectors are $l^x = 0$ and $u^x = \infty$. Moreover, it is possible to specify several sets of bound vectors. A record in this section has the form

```
?? [name]    [vname1]    [value1]
```

where the requirements for each field are:

Field	Starting Position	Max Width	Required	Description
??	2	2	Yes	Bound key
[name]	5	8	Yes	Name of the BOUNDS vector
[vname1]	15	8	Yes	Variable name
[value1]	25	12	No	Numerical value

Hence, a record in the BOUNDS section has the following interpretation: [name] is the name of the bound vector and [vname1] is the name of the variable which bounds are modified by the record. ?? and [value1] are used to modify the bound vectors according to the following table:

??	l_j^x	u_j^x	Made integer (added to $\mathcal{J}$)
FR	$-\infty$	∞	No
FX	v_1	v_1	No
LO	v_1	unchanged	No
MI	$-\infty$	unchanged	No
PL	unchanged	∞	No
UP	unchanged	v_1	No
BV	0	1	Yes
LI	$\lceil v_1 \rceil$	unchanged	Yes
UI	unchanged	$\lfloor v_1 \rfloor$	Yes

v_1 is the value specified by [value1].

11.2.4 CSECTION (optional)

The purpose of the CSECTION is to specify the constraint

$$x \in \mathcal{K}.$$

in (11.2). It is assumed that $\mathcal{K}$ satisfies the following requirements. Let

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

be vectors comprised of parts of the decision variables x so that each decision variable is a member of exactly **one** vector x^t, for example

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

Next define

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

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

- $\mathbb{R}$ set:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} \right\}.$$

- Quadratic cone:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\}. \tag{11.3}$$

- Rotated quadratic cone:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : 2x_1 x_2 \geq \sum_{j=3}^{n^t} x_j^2, \quad x_1, x_2 \geq 0 \right\}. \tag{11.4}$$

In general, only quadratic and rotated quadratic cones are specified in the MPS file whereas membership of the $\mathbb{R}$ set is not. If a variable is not a member of any other cone then it is assumed to be a member of an $\mathbb{R}$ cone.

Next, let us study an example. Assume that the quadratic cone

$$x_4 \geq \sqrt{x_5^2 + x_8^2}$$

and the rotated quadratic cone

$$x_3 x_7 \geq x_1^2 + x_0^2, \quad x_3, x_7 \geq 0,$$

should be specified in the MPS file. One CSECTION is required for each cone and they are specified as follows:

```
*       1         2         3         4         5         6
*23456789012345678901234567890123456789012345678901234567890
CSECTION      konea       0.0          QUAD
x4
x5
x8
CSECTION      koneb       0.0          RQUAD
x7
x3
x1
x0
```

This first CSECTION specifies the cone (11.3) which is given the name konea. This is a quadratic cone which is specified by the keyword QUAD in the CSECTION header. The 0.0 value in the CSECTION header is not used by the QUAD cone.

The second CSECTION specifies the rotated quadratic cone (11.4). Please note the keyword RQUAD in the CSECTION which is used to specify that the cone is a rotated quadratic cone instead of a quadratic cone. The 0.0 value in the CSECTION header is not used by the RQUAD cone.

In general, a CSECTION header has the format

```
CSECTION      [kname1]    [value1]     [ktype]
```

where the requirement for each field are as follows:

Field	Starting Position	Max Width	Required	Description
[kname1]	5	8	Yes	Name of the cone
[value1]	15	12	No	Cone parameter
[ktype]	25		Yes	Type of the cone.

The possible cone type keys are:

Cone type key	Members	Interpretation.
QUAD	≤ 1	Quadratic cone i.e. (11.3).
RQUAD	≤ 2	Rotated quadratic cone i.e. (11.4).

Please note that a quadratic cone must have at least one member whereas a rotated quadratic cone must have at least two members. A record in the CSECTION has the format

```
[vname1]
```

where the requirements for each field are

Field	Starting Position	Max Width	required	Description
[vname1]	2	8	Yes	A valid variable name

The most important restriction with respect to the CSECTION is that a variable must occur in only one CSECTION.

11.2.5 ENDATA

This keyword denotes the end of the MPS file.

11.2.6 Integer Variables

Using special bound keys in the BOUNDS section it is possible to specify that some or all of the variables should be integer-constrained i.e. be members of $\mathcal{J}$. However, an alternative method is available.

This method is available only for backward compatibility and we recommend that it is not used. This method requires that markers are placed in the COLUMNS section as in the example:

```
COLUMNS
x1        obj       -10.0         c1        0.7
x1        c2        0.5           c3        1.0
x1        c4        0.1
* Start of integer-constrained variables.
MARK000   'MARKER'                'INTORG'
x2        obj       -9.0          c1        1.0
x2        c2        0.8333333333  c3        0.66666667
x2        c4        0.25
x3        obj       1.0           c6        2.0
MARK001   'MARKER'                'INTEND'
```

- End of integer-constrained variables.

Please note that special marker lines are used to indicate the start and the end of the integer variables. Furthermore be aware of the following

- IMPORTANT: All variables between the markers are assigned a default lower bound of 0 and a default upper bound of 1. **This may not be what is intended.** If it is not intended, the correct bounds should be defined in the `BOUNDS` section of the MPS formatted file.
- **MOSEK** ignores field 1, i.e. `MARK0001` and `MARK001`, however, other optimization systems require them.
- Field 2, i.e. `MARKER`, must be specified including the single quotes. This implies that no row can be assigned the name `MARKER`.
- Field 3 is ignored and should be left blank.
- Field 4, i.e. `INTORG` and `INTEND`, must be specified.
- It is possible to specify several such integer marker sections within the `COLUMNS` section.

11.2.7 General Limitations

- An MPS file should be an ASCII file.

11.2.8 Interpretation of the MPS Format

Several issues related to the MPS format are not well-defined by the industry standard. However, **MOSEK** uses the following interpretation:

- If a matrix element in the `COLUMNS` section is specified multiple times, then the multiple entries are added together.
- If a matrix element in a `QSECTION` section is specified multiple times, then the multiple entries are added together.

11.2.9 The Free MPS Format

MOSEK supports a free format variation of the MPS format. The free format is similar to the MPS file format but less restrictive, e.g. it allows longer names. However, it also presents two main limitations:

- A name must not contain any blanks.
- By default a line in the MPS file must not contain more than 1024 characters. However, by modifying the parameter *MSK_IPAR_READ_MPS_WIDTH* an arbitrary large line width will be accepted.

To use the free MPS format instead of the default MPS format the **MOSEK** parameter *MSK_IPAR_READ_MPS_FORMAT* should be changed.

11.3 The OPF Format

The *Optimization Problem Format (OPF)* is an alternative to LP and MPS files for specifying optimization problems. It is row-oriented, inspired by the CPLEX LP format.

Apart from containing objective, constraints, bounds etc. it may contain complete or partial solutions, comments and extra information relevant for solving the problem. It is designed to be easily read and modified by hand and to be forward compatible with possible future extensions.

Intended use

The OPF file format is meant to replace several other files:

- The LP file format: Any problem that can be written as an LP file can be written as an OPF file too; furthermore it naturally accommodates ranged constraints and variables as well as arbitrary characters in names, fixed expressions in the objective, empty constraints, and conic constraints.
- Parameter files: It is possible to specify integer, double and string parameters along with the problem (or in a separate OPF file).
- Solution files: It is possible to store a full or a partial solution in an OPF file and later reload it.

11.3.1 The File Format

The format uses tags to structure data. A simple example with the basic sections may look like this:

```
[comment]
This is a comment. You may write almost anything here...
[/comment]

# This is a single-line comment.

[objective min 'myobj']
x + 3 y + x^2 + 3 y^2 + z + 1
[/objective]

[constraints]
[con 'con01'] 4 &<= x + y  [/con]
[/constraints]

[bounds]
[b] -10 &<= x,y &<= 10  [/b]

[cone quad] x,y,z [/cone]
[/bounds]
```

A scope is opened by a tag of the form `[tag]` and closed by a tag of the form `[/tag]`. An opening tag may accept a list of unnamed and named arguments, for examples:

```
[tag value] tag with one unnamed argument [/tag]
[tag arg=value] tag with one named argument in quotes [/tag]
```

Unnamed arguments are identified by their order, while named arguments may appear in any order, but never before an unnamed argument. The `value` can be a quoted, single-quoted or double-quoted text string, i.e.

```
[tag 'value']     single-quoted value [/tag]
[tag arg='value'] single-quoted value [/tag]
[tag "value"]     double-quoted value [/tag]
[tag arg="value"] double-quoted value [/tag]
```

Sections

The recognized tags are

`[comment]`

A comment section. This can contain *almost* any text: Between single quotes (') or double quotes (") any text may appear. Outside quotes the markup characters ([and]) must be prefixed by backslashes. Both single and double quotes may appear alone or inside a pair of quotes if it is prefixed by a backslash.

`[objective]`

The objective function: This accepts one or two parameters, where the first one (in the above example `min`) is either `min` or `max` (regardless of case) and defines the objective sense, and the second one (above `myobj`), if present, is the objective name. The section may contain linear and quadratic expressions. If several objectives are specified, all but the last are ignored.

`[constraints]`

This does not directly contain any data, but may contain the subsection `con` defining a linear constraint.

`[con]` defines a single constraint; if an argument is present (`[con NAME]`) this is used as the name of the constraint, otherwise it is given a null-name. The section contains a constraint definition written as linear and quadratic expressions with a lower bound, an upper bound, with both or with an equality. Examples:

```
[constraints]
[con 'con1'] 0 <= x + y       [/con]
[con 'con2'] 0 >= x + y       [/con]
[con 'con3'] 0 <= x + y <= 10 [/con]
[con 'con4']      x + y  = 10 [/con]
[/constraints]
```

Constraint names are unique. If a constraint is specified which has the same name as a previously defined constraint, the new constraint replaces the existing one.

`[bounds]`

This does not directly contain any data, but may contain the subsections `b` (linear bounds on variables) and `cone` (quadratic cone).

`[b]`. Bound definition on one or several variables separated by comma (`,`). An upper or lower bound on a variable replaces any earlier defined bound on that variable. If only one bound (upper or lower) is given only this bound is replaced. This means that upper and lower bounds can be specified separately. So the OPF bound definition:

```
[b]  x,y >= -10  [/b]
[b]  x,y <= 10   [/b]
```

results in the bound $-10 \leq x, y \leq 10$.

`[cone]`. currently supports the *quadratic cone* and the *rotated quadratic cone*.

A conic constraint is defined as a set of variables which belong to a single unique cone.

- A quadratic cone of n variables $x_1, \ldots, x_n$ defines a constraint of the form

$$x_1^2 > \sum_{i=2}^{n} x_i^2.$$

- A rotated quadratic cone of n variables $x_1, \ldots, x_n$ defines a constraint of the form

$$x_1 x_2 > \sum_{i=3}^{n} x_i^2.$$

A [bounds]-section example:

```
[bounds]
[b]  0 <= x,y <= 10  [/b] # ranged bound
[b] 10 >= x,y >=  0  [/b] # ranged bound
[b]  0 <= x,y <= inf [/b] # using inf
[b]      x,y free    [/b] # free variables
# Let (x,y,z,w) belong to the cone K
[cone quad]  x,y,z,w  [/cone] # quadratic cone
[cone rquad] x,y,z,w  [/cone] # rotated quadratic cone
[/bounds]
```

By default all variables are free.

[variables]

This defines an ordering of variables as they should appear in the problem. This is simply a space-separated list of variable names.

[integer]

This contains a space-separated list of variables and defines the constraint that the listed variables must be integer values.

[hints]

This may contain only non-essential data; for example estimates of the number of variables, constraints and non-zeros. Placed before all other sections containing data this may reduce the time spent reading the file.

In the hints section, any subsection which is not recognized by **MOSEK** is simply ignored. In this section a hint in a subsection is defined as follows:

```
[hint ITEM] value [/hint]
```

where ITEM may be replaced by numvar (number of variables), numcon (number of linear/quadratic constraints), numanz (number of linear non-zeros in constraints) and numqnz (number of quadratic non-zeros in constraints).

[solutions]

This section can contain a set of full or partial solutions to a problem. Each solution must be specified using a [solution]-section, i.e.

```
[solutions]
[solution]...[/solution] #solution 1
[solution]...[/solution] #solution 2
#other solutions....
[solution]...[/solution] #solution n
[/solutions]
```

Note that a [solution]-section must be always specified inside a [solutions]-section. The syntax of a [solution]-section is the following:

```
[solution SOLTYPE status=STATUS]...[/solution]
```

where SOLTYPE is one of the strings

- interior, a non-basic solution,
- basic, a basic solution,
- integer, an integer solution,

and STATUS is one of the strings

- UNKNOWN,
- OPTIMAL,
- INTEGER_OPTIMAL,
- PRIM_FEAS,
- DUAL_FEAS,
- PRIM_AND_DUAL_FEAS,
- NEAR_OPTIMAL,
- NEAR_PRIM_FEAS,
- NEAR_DUAL_FEAS,
- NEAR_PRIM_AND_DUAL_FEAS,
- PRIM_INFEAS_CER,
- DUAL_INFEAS_CER,
- NEAR_PRIM_INFEAS_CER,
- NEAR_DUAL_INFEAS_CER,
- NEAR_INTEGER_OPTIMAL.

Most of these values are irrelevant for input solutions; when constructing a solution for simplex hot-start or an initial solution for a mixed integer problem the safe setting is UNKNOWN.

A [solution]-section contains [con] and [var] sections. Each [con] and [var] section defines solution information for a single variable or constraint, specified as list of KEYWORD/value pairs, in any order, written as

```
KEYWORD=value
```

Allowed keywords are as follows:

- sk. The status of the item, where the value is one of the following strings:
  - LOW, the item is on its lower bound.
  - UPR, the item is on its upper bound.
  - FIX, it is a fixed item.
  - BAS, the item is in the basis.

  - SUPBAS, the item is super basic.
  - UNK, the status is unknown.
  - INF, the item is outside its bounds (infeasible).
- lvl Defines the level of the item.
- sl Defines the level of the dual variable associated with its lower bound.
- su Defines the level of the dual variable associated with its upper bound.
- sn Defines the level of the variable associated with its cone.
- y Defines the level of the corresponding dual variable (for constraints only).

A [var] section should always contain the items sk, lvl, sl and su. Items sl and su are not required for integer solutions.

A [con] section should always contain sk, lvl, sl, su and y.

An example of a solution section

```
[solution basic status=UNKNOWN]
[var x0] sk=LOW    lvl=5.0      [/var]
[var x1] sk=UPR    lvl=10.0     [/var]
[var x2] sk=SUPBAS lvl=2.0  sl=1.5 su=0.0 [/var]

[con c0] sk=LOW    lvl=3.0 y=0.0 [/con]
[con c0] sk=UPR    lvl=0.0 y=5.0 [/con]
[/solution]
```

- [vendor] This contains solver/vendor specific data. It accepts one argument, which is a vendor ID – for **MOSEK** the ID is simply mosek – and the section contains the subsection parameters defining solver parameters. When reading a vendor section, any unknown vendor can be safely ignored. This is described later.

Comments using the # may appear anywhere in the file. Between the # and the following line-break any text may be written, including markup characters.

Numbers

Numbers, when used for parameter values or coefficients, are written in the usual way by the printf function. That is, they may be prefixed by a sign (+ or -) and may contain an integer part, decimal part and an exponent. The decimal point is always . (a dot). Some examples are

```
1
1.0
.0
1.
1e10
1e+10
1e-10
```

Some *invalid* examples are

```
e10   # invalid, must contain either integer or decimal part
.     # invalid
.e10  # invalid
```

More formally, the following standard regular expression describes numbers as used:

```
[+|-]?([0-9]+[.][0-9]*|[.][0-9]+)([eE][+|-]?[0-9]+)?
```

Names

Variable names, constraint names and objective name may contain arbitrary characters, which in some cases must be enclosed by quotes (single or double) that in turn must be preceded by a backslash. Unquoted names must begin with a letter (a-z or A-Z) and contain only the following characters: the letters a-z and A-Z, the digits 0-9, braces ({ and }) and underscore (_).

Some examples of legal names:

```
an_unquoted_name
another_name{123}
'single quoted name'
"double quoted name"
"name with \\"quote\\" in it"
"name with []s in it"
```

11.3.2 Parameters Section

In the vendor section solver parameters are defined inside the parameters subsection. Each parameter is written as

```
[p PARAMETER_NAME] value [/p]
```

where PARAMETER_NAME is replaced by a **MOSEK** parameter name, usually of the form MSK_IPAR_..., MSK_DPAR_... or MSK_SPAR_..., and the value is replaced by the value of that parameter; both integer values and named values may be used. Some simple examples are

```
[vendor mosek]
[parameters]
[p MSK_IPAR_OPF_MAX_TERMS_PER_LINE] 10     [/p]
[p MSK_IPAR_OPF_WRITE_PARAMETERS]   MSK_ON [/p]
[p MSK_DPAR_DATA_TOL_BOUND_INF]     1.0e18 [/p]
[/parameters]
[/vendor]
```

11.3.3 Writing OPF Files from MOSEK

To write an OPF file set the parameter *MSK_IPAR_WRITE_DATA_FORMAT* to *MSK_DATA_FORMAT_OP* as this ensures that OPF format is used.

Then modify the following parameters to define what the file should contain:

MSK_IPAR_OPF_WRITE_SOL_BAS	Include basic solution, if defined.
MSK_IPAR_OPF_WRITE_SOL_ITG	Include integer solution, if defined.
MSK_IPAR_OPF_WRITE_SOL_ITR	Include interior solution, if defined.
MSK_IPAR_OPF_WRITE_SOLUTIONS	Include solutions if they are defined. If this is off, no solutions are included.
MSK_IPAR_OPF_WRITE_HEADER	Include a small header with comments.
MSK_IPAR_OPF_WRITE_PROBLEM	Include the problem itself — objective, constraints and bounds.
MSK_IPAR_OPF_WRITE_PARAMETERS	Include all parameter settings.
MSK_IPAR_OPF_WRITE_HINTS	Include hints about the size of the problem.

11.3.4 Examples

This section contains a set of small examples written in OPF and describing how to formulate linear, quadratic and conic problems.

Linear Example lo1.opf

Consider the example:

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

having the bounds

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

In the OPF format the example is displayed as shown in Listing 11.1.

Listing 11.1: Example of an OPF file for a linear problem.

```
[comment]
  The lo1 example in OPF format
[/comment]

[hints]
  [hint NUMVAR] 4 [/hint]
  [hint NUMCON] 3 [/hint]
  [hint NUMANZ] 9 [/hint]
[/hints]

[variables disallow_new_variables]
  x1 x2 x3 x4
[/variables]

[objective maximize 'obj']
   3 x1 + x2 + 5 x3 + x4
[/objective]

[constraints]
  [con 'c1'] 3 x1 +   x2 + 2 x3         = 30 [/con]
  [con 'c2'] 2 x1 +   x2 + 3 x3 +   x4 >= 15 [/con]
  [con 'c3']        2 x2        + 3 x4 <= 25 [/con]
[/constraints]

[bounds]
  [b] 0 <= * [/b]
  [b] 0 <= x2 <= 10 [/b]
[/bounds]
```

Quadratic Example qo1.opf

An example of a quadratic optimization problem is

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

This can be formulated in opf as shown below.

Listing 11.2: Example of an OPF file for a quadratic problem.

```
[comment]
  The qo1 example in OPF format
[/comment]

[hints]
  [hint NUMVAR] 3 [/hint]
  [hint NUMCON] 1 [/hint]
  [hint NUMANZ] 3 [/hint]
  [hint NUMQNZ] 4 [/hint]
[/hints]

[variables disallow_new_variables]
  x1 x2 x3
[/variables]

[objective minimize 'obj']
  # The quadratic terms are often written with a factor of 1/2 as here,
  # but this is not required.

   - x2 + 0.5 ( 2.0 x1 ^ 2 - 2.0 x3 * x1 + 0.2 x2 ^ 2 + 2.0 x3 ^ 2 )
[/objective]

[constraints]
  [con 'c1'] 1.0 <= x1 + x2 + x3 [/con]
[/constraints]

[bounds]
  [b] 0 <= * [/b]
[/bounds]
```

Conic Quadratic Example `cqo1.opf`

Consider the example:

$$\begin{array}{lccl} \text{minimize} & x_3 + x_4 + x_5 & & \\ \text{subject to} & x_0 + x_1 + 2x_2 & = & 1, \\ & x_0, x_1, x_2 & \geq & 0, \\ & x_3 \geq \sqrt{x_0^2 + x_1^2}, & & \\ & 2x_4 x_5 \geq x_2^2. & & \end{array}$$

Please note that the type of the cones is defined by the parameter to `[cone ...]`; the content of the `cone`-section is the names of variables that belong to the cone. The resulting OPF file is in Listing 11.3.

Listing 11.3: Example of an OPF file for a conic quadratic problem.

```
[comment]
  The cqo1 example in OPF format.
[/comment]

[hints]
  [hint NUMVAR] 6 [/hint]
  [hint NUMCON] 1 [/hint]
  [hint NUMANZ] 3 [/hint]
[/hints]

[variables disallow_new_variables]
  x1 x2 x3 x4 x5 x6
[/variables]
```

```
[objective minimize 'obj']
   x4 + x5 + x6
[/objective]

[constraints]
  [con 'c1']  x1 + x2 + 2e+00 x3 = 1e+00 [/con]
[/constraints]

[bounds]
  # We let all variables default to the positive orthant
  [b] 0 <= * [/b]

  # ...and change those that differ from the default
  [b] x4,x5,x6 free [/b]

  # Define quadratic cone: x4 >= sqrt( x1^2 + x2^2 )
  [cone quad 'k1'] x4, x1, x2 [/cone]

  # Define rotated quadratic cone: 2 x5 x6 >= x3^2
  [cone rquad 'k2'] x5, x6, x3 [/cone]
[/bounds]
```

Mixed Integer Example `milo1.opf`

Consider the mixed integer problem:

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

This can be implemented in `OPF` with the file in Listing 11.4.

Listing 11.4: Example of an OPF file for a mixed-integer linear problem.

```
[comment]
  The milo1 example in OPF format
[/comment]

[hints]
  [hint NUMVAR] 2 [/hint]
  [hint NUMCON] 2 [/hint]
  [hint NUMANZ] 4 [/hint]
[/hints]

[variables disallow_new_variables]
  x1 x2
[/variables]

[objective maximize 'obj']
   x1 + 6.4e-1 x2
[/objective]

[constraints]
  [con 'c1'] 5e+1 x1 + 3.1e+1 x2 <= 2.5e+2 [/con]
  [con 'c2'] -4 <= 3 x1 - 2 x2 [/con]
[/constraints]

[bounds]
  [b] 0 <= * [/b]
[/bounds]
```

```
[integer]
  x1 x2
[/integer]
```

11.4 The CBF Format

This document constitutes the technical reference manual of the *Conic Benchmark Format* with file extension: `.cbf` or `.CBF`. It unifies linear, second-order cone (also known as conic quadratic) and semidefinite optimization with mixed-integer variables. The format has been designed with benchmark libraries in mind, and therefore focuses on compact and easily parsable representations. The problem structure is separated from the problem data, and the format moreover facilitates benchmarking of hotstart capability through sequences of changes.

11.4.1 How Instances Are Specified

This section defines the spectrum of conic optimization problems that can be formulated in terms of the keywords of the CBF format.

In the CBF format, conic optimization problems are considered in the following form:

$$\begin{array}{lll} \min / \max & g^{obj} & \\ & g_i \in \mathcal{K}_i, & i \in \mathcal{I}, \\ \text{s.t.} & G_i \in \mathcal{K}_i, & i \in \mathcal{I}^{PSD}, \\ & x_j \in \mathcal{K}_j, & j \in \mathcal{J}, \\ & \overline{X}_j \in \mathcal{K}_j, & j \in \mathcal{J}^{PSD}. \end{array} \tag{11.5}$$

- **Variables** are either scalar variables, x_j for $j \in \mathcal{J}$, or variables, $\overline{X}_j$ for $j \in \mathcal{J}^{PSD}$. Scalar variables can also be declared as integer.
- **Constraints** are affine expressions of the variables, either scalar-valued g_i for $i \in \mathcal{I}$, or matrix-valued G_i for $i \in \mathcal{I}^{PSD}$

$$\begin{aligned} g_i &= \sum_{j \in \mathcal{J}^{PSD}} \langle F_{ij}, X_j \rangle + \sum_{j \in \mathcal{J}} a_{ij} x_j + b_i, \\ G_i &= \sum_{j \in \mathcal{J}} x_j H_{ij} + D_i. \end{aligned}$$

- The **objective function** is a scalar-valued affine expression of the variables, either to be minimized or maximized. We refer to this expression as g^{obj}

$$g^{obj} = \sum_{j \in \mathcal{J}^{PSD}} \langle F_j^{obj}, X_j \rangle + \sum_{j \in \mathcal{J}} a_j^{obj} x_j + b^{obj}.$$

CBF format can represent the following cones $\mathcal{K}$:

- **Free domain** - A cone in the linear family defined by

$$\{x \in \mathbb{R}^n\}, \text{ for } n \geq 1.$$

- **Positive orthant** - A cone in the linear family defined by

$$\{x \in \mathbb{R}^n \mid x_j \geq 0 \text{ for } j = 1, \ldots, n\}, \text{ for } n \geq 1.$$

- **Negative orthant** - A cone in the linear family defined by

$$\{x \in \mathbb{R}^n \mid x_j \leq 0 \text{ for } j = 1, \ldots, n\}, \text{ for } n \geq 1.$$

- **Fixpoint zero** - A cone in the linear family defined by

$$\{x \in \mathbb{R}^n \mid x_j = 0 \text{ for } j = 1, \ldots, n\}, \text{ for } n \geq 1.$$

- **Quadratic cone** - A cone in the second-order cone family defined by

$$\left\{ \begin{pmatrix} p \\ x \end{pmatrix} \in \mathbb{R} \times \mathbb{R}^{n-1},\ p^2 \geq x^T x,\ p \geq 0 \right\}, \text{ for } n \geq 2.$$

- **Rotated quadratic cone** - A cone in the second-order cone family defined by

$$\left\{ \begin{pmatrix} p \\ q \\ x \end{pmatrix} \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^{n-2},\ 2pq \geq x^T x,\ p \geq 0,\ q \geq 0 \right\}, \text{ for } n \geq 3.$$

11.4.2 The Structure of CBF Files

This section defines how information is written in the CBF format, without being specific about the type of information being communicated.

All information items belong to exactly one of the three groups of information. These information groups, and the order they must appear in, are:

1. File format.
2. Problem structure.
3. Problem data.

The first group, file format, provides information on how to interpret the file. The second group, problem structure, provides the information needed to deduce the type and size of the problem instance. Finally, the third group, problem data, specifies the coefficients and constants of the problem instance.

Information items

The format is composed as a list of information items. The first line of an information item is the `KEYWORD`, revealing the type of information provided. The second line - of some keywords only - is the `HEADER`, typically revealing the size of information that follows. The remaining lines are the `BODY` holding the actual information to be specified.

```
KEYWORD
BODY

KEYWORD
HEADER
BODY
```

The `KEYWORD` determines how each line in the `HEADER` and `BODY` is structured. Moreover, the number of lines in the `BODY` follows either from the `KEYWORD`, the `HEADER`, or from another information item required to precede it.

Embedded hotstart-sequences

A sequence of problem instances, based on the same problem structure, is within a single file. This is facilitated via the `CHANGE` within the problem data information group, as a separator between the information items of each instance. The information items following a `CHANGE` keyword are appending to, or changing (e.g., setting coefficients back to their default value of zero), the problem data of the preceding instance.

The sequence is intended for benchmarking of hotstart capability, where the solvers can reuse their internal state and solution (subject to the achieved accuracy) as warmpoint for the succeeding instance. Whenever this feature is unsupported or undesired, the keyword `CHANGE` should be interpreted as the end of file.

File encoding and line width restrictions

The format is based on the US-ASCII printable character set with two extensions as listed below. Note, by definition, that none of these extensions can be misinterpreted as printable US-ASCII characters:

- A line feed marks the end of a line, carriage returns are ignored.
- Comment-lines may contain unicode characters in UTF-8 encoding.

The line width is restricted to 512 bytes, with 3 bytes reserved for the potential carriage return, line feed and null-terminator.

Integers and floating point numbers must follow the ISO C decimal string representation in the standard C locale. The format does not impose restrictions on the magnitude of, or number of significant digits in numeric data, but the use of 64-bit integers and 64-bit IEEE 754 floating point numbers should be sufficient to avoid loss of precision.

Comment-line and whitespace rules

The format allows single-line comments respecting the following rule:

- Lines having first byte equal to '`#`' (US-ASCII 35) are comments, and should be ignored. Comments are only allowed between information items.

Given that a line is not a comment-line, whitespace characters should be handled according to the following rules:

- Leading and trailing whitespace characters should be ignored.
  - The seperator between multiple pieces of information on one line, is either one or more whitespace characters.
- Lines containing only whitespace characters are empty, and should be ignored. Empty lines are only allowed between information items.

11.4.3 Problem Specification

The problem structure

The problem structure defines the objective sense, whether it is minimization and maximization. It also defines the index sets, $\mathcal{J}$, $\mathcal{J}^{PSD}$, $\mathcal{I}$ and $\mathcal{I}^{PSD}$, which are all numbered from zero, $\{0, 1, \ldots\}$, and empty until explicitly constructed.

- **Scalar variables** are constructed in vectors restricted to a conic domain, such as $(x_0, x_1) \in \mathbb{R}^2_+$, $(x_2, x_3, x_4) \in \mathcal{Q}^3$, etc. In terms of the Cartesian product, this generalizes to

$$x \in \mathcal{K}_1^{n_1} \times \mathcal{K}_2^{n_2} \times \cdots \times \mathcal{K}_k^{n_k}$$

which in the CBF format becomes:

```
VAR
n k
K1 n1
K2 n2
...
Kk nk
```

where $\sum_i n_i = n$ is the total number of scalar variables. The list of supported cones is found in Table 11.3. Integrality of scalar variables can be specified afterwards.

- **PSD variables** are constructed one-by-one. That is, $X_j \succeq \mathbf{0}^{n_j \times n_j}$ for $j \in \mathcal{J}^{PSD}$, constructs a matrix-valued variable of size $n_j \times n_j$ restricted to be symmetric positive semidefinite. In the CBF format, this list of constructions becomes:

```
PSDVAR
N
n1
n2
...
nN
```

where N is the total number of PSD variables.

- **Scalar constraints** are constructed in vectors restricted to a conic domain, such as $(g_0, g_1) \in \mathbb{R}^2_+$, $(g_2, g_3, g_4) \in \mathcal{Q}^3$, etc. In terms of the Cartesian product, this generalizes to

$$g \in \mathcal{K}_1^{m_1} \times \mathcal{K}_2^{m_2} \times \cdots \times \mathcal{K}_k^{m_k}$$

which in the CBF format becomes:

```
CON
m k
K1 m1
K2 m2
..
Kk mk
```

where $\sum_i m_i = m$ is the total number of scalar constraints. The list of supported cones is found in Table 11.3.

- **PSD constraints** are constructed one-by-one. That is, $G_i \succeq \mathbf{0}^{m_i \times m_i}$ for $i \in \mathcal{I}^{PSD}$, constructs a matrix-valued affine expressions of size $m_i \times m_i$ restricted to be symmetric positive semidefinite. In the CBF format, this list of constructions becomes

```
PSDCON
M
m1
m2
..
mM
```

where M is the total number of PSD constraints.

With the objective sense, variables (with integer indications) and constraints, the definitions of the many affine expressions follow in problem data.

Problem data

The problem data defines the coefficients and constants of the affine expressions of the problem instance. These are considered zero until explicitly defined, implying that instances with no keywords from this information group are, in fact, valid. Duplicating or conflicting information is a failure to comply with the standard. Consequently, two coefficients written to the same position in a matrix (or to transposed positions in a symmetric matrix) is an error.

The affine expressions of the objective, g^{obj}, of the scalar constraints, g_i, and of the PSD constraints, G_i, are defined separately. The following notation uses the standard trace inner product for matrices, $\langle X, Y\rangle = \sum_{i,j} X_{ij}Y_{ij}$.

- The affine expression of the objective is defined as

$$g^{obj} = \sum_{j\in\mathcal{J}^{PSD}} \langle F_j^{obj}, X_j\rangle + \sum_{j\in\mathcal{J}} a_j^{obj} x_j + b^{obj},$$

  in terms of the symmetric matrices, F_j^{obj}, and scalars, a_j^{obj} and b^{obj}.

- The affine expressions of the scalar constraints are defined, for $i \in \mathcal{I}$, as

$$g_i = \sum_{j\in\mathcal{J}^{PSD}} \langle F_{ij}, X_j\rangle + \sum_{j\in\mathcal{J}} a_{ij} x_j + b_i,$$

  in terms of the symmetric matrices, F_{ij}, and scalars, a_{ij} and b_i.

- The affine expressions of the PSD constraints are defined, for $i \in \mathcal{I}^{PSD}$, as

$$G_i = \sum_{j\in\mathcal{J}} x_j H_{ij} + D_i,$$

  in terms of the symmetric matrices, H_{ij} and D_i.

List of cones

The format uses an explicit syntax for symmetric positive semidefinite cones as shown above. For scalar variables and constraints, constructed in vectors, the supported conic domains and their minimum sizes are given as follows.

Table 11.3: Cones available in the CBF format

Name	CBF keyword	Cone family
Free domain	`F`	linear
Positive orthant	`L+`	linear
Negative orthant	`L-`	linear
Fixpoint zero	`L=`	linear
Quadratic cone	`Q`	second-order
Rotated quadratic cone	`QR`	second-order

11.4.4 File Format Keywords

VER

Description: The version of the Conic Benchmark Format used to write the file.

`HEADER`: None

`BODY`: One line formatted as:

```
INT
```

This is the version number.

Must appear exactly once in a file, as the first keyword.

OBJSENSE

Description: Define the objective sense.

`HEADER`: None

`BODY`: One line formatted as:

```
STR
```

having `MIN` indicates minimize, and `MAX` indicates maximize. Capital letters are required.

Must appear exactly once in a file.

PSDVAR

Description: Construct the PSD variables.

`HEADER`: One line formatted as:

```
INT
```

This is the number of PSD variables in the problem.

`BODY`: A list of lines formatted as:

```
INT
```

This indicates the number of rows (equal to the number of columns) in the matrix-valued PSD variable. The number of lines should match the number stated in the header.

VAR

Description: Construct the scalar variables.

`HEADER`: One line formatted as:

```
INT INT
```

This is the number of scalar variables, followed by the number of conic domains they are restricted to.

`BODY`: A list of lines formatted as:

```
STR INT
```

This indicates the cone name (see Table 11.3), and the number of scalar variables restricted to this cone. These numbers should add up to the number of scalar variables stated first in the header. The number of lines should match the second number stated in the header.

INT

Description: Declare integer requirements on a selected subset of scalar variables.

`HEADER`: one line formatted as:

```
INT
```

This is the number of integer scalar variables in the problem.

`BODY`: a list of lines formatted as:

```
INT
```

This indicates the scalar variable index $j \in \mathcal{J}$. The number of lines should match the number stated in the header.

Can only be used after the keyword `VAR`.

PSDCON

Description: Construct the PSD constraints.

`HEADER`: One line formatted as:

```
INT
```

This is the number of PSD constraints in the problem.

`BODY`: A list of lines formatted as:

```
INT
```

This indicates the number of rows (equal to the number of columns) in the matrix-valued affine expression of the PSD constraint. The number of lines should match the number stated in the header.

Can only be used after these keywords: `PSDVAR`, `VAR`.

CON

Description: Construct the scalar constraints.

`HEADER`: One line formatted as:

```
INT INT
```

This is the number of scalar constraints, followed by the number of conic domains they restrict to.

`BODY`: A list of lines formatted as:

```
STR INT
```

This indicates the cone name (see Table 11.3), and the number of affine expressions restricted to this cone. These numbers should add up to the number of scalar constraints stated first in the header. The number of lines should match the second number stated in the header.

Can only be used after these keywords: `PSDVAR`, `VAR`

OBJFCOORD

Description: Input sparse coordinates (quadruplets) to define the symmetric matrices F_j^{obj}, as used in the objective.

`HEADER`: One line formatted as:

```
INT
```

This is the number of coordinates to be specified.

`BODY`: A list of lines formatted as:

```
INT INT INT REAL
```

This indicates the PSD variable index $j \in \mathcal{J}^{PSD}$, the row index, the column index and the coefficient value. The number of lines should match the number stated in the header.

OBJACOORD

Description: Input sparse coordinates (pairs) to define the scalars, a_j^{obj}, as used in the objective.

`HEADER`: One line formatted as:

```
INT
```

This is the number of coordinates to be specified.

`BODY`: A list of lines formatted as:

```
INT REAL
```

This indicates the scalar variable index $j \in \mathcal{J}$ and the coefficient value. The number of lines should match the number stated in the header.

OBJBCOORD

Description: Input the scalar, b^{obj}, as used in the objective.

`HEADER`: None.

`BODY`: One line formatted as:

```
REAL
```

This indicates the coefficient value.

FCOORD

Description: Input sparse coordinates (quintuplets) to define the symmetric matrices, F_{ij}, as used in the scalar constraints.

`HEADER`: One line formatted as:

```
INT
```

This is the number of coordinates to be specified.

`BODY`: A list of lines formatted as:

```
INT INT INT INT REAL
```

This indicates the scalar constraint index $i \in \mathcal{I}$, the PSD variable index $j \in \mathcal{J}^{PSD}$, the row index, the column index and the coefficient value. The number of lines should match the number stated in the header.

ACOORD

Description: Input sparse coordinates (triplets) to define the scalars, a_{ij}, as used in the scalar constraints.

`HEADER`: One line formatted as:

```
INT
```

This is the number of coordinates to be specified.

BODY: A list of lines formatted as:

```
INT INT REAL
```

This indicates the scalar constraint index $i \in \mathcal{I}$, the scalar variable index $j \in \mathcal{J}$ and the coefficient value. The number of lines should match the number stated in the header.

BCOORD

Description: Input sparse coordinates (pairs) to define the scalars, b_i, as used in the scalar constraints.

HEADER: One line formatted as:

```
INT
```

This is the number of coordinates to be specified.

BODY: A list of lines formatted as:

```
INT REAL
```

This indicates the scalar constraint index $i \in \mathcal{I}$ and the coefficient value. The number of lines should match the number stated in the header.

HCOORD

Description: Input sparse coordinates (quintuplets) to define the symmetric matrices, H_{ij}, as used in the PSD constraints.

HEADER: One line formatted as:

```
INT
```

This is the number of coordinates to be specified.

BODY: A list of lines formatted as

```
INT INT INT INT REAL
```

This indicates the PSD constraint index $i \in \mathcal{I}^{PSD}$, the scalar variable index $j \in \mathcal{J}$, the row index, the column index and the coefficient value. The number of lines should match the number stated in the header.

DCOORD

Description: Input sparse coordinates (quadruplets) to define the symmetric matrices, D_i, as used in the PSD constraints.

HEADER: One line formatted as

```
INT
```

This is the number of coordinates to be specified.

BODY: A list of lines formatted as:

```
INT INT INT REAL
```

This indicates the PSD constraint index $i \in \mathcal{I}^{PSD}$, the row index, the column index and the coefficient value. The number of lines should match the number stated in the header.

CHANGE

Start of a new instance specification based on changes to the previous. Can be interpreted as the end of file when the hotstart-sequence is unsupported or undesired.

`BODY`: None

`Header`: None

11.4.5 CBF Format Examples

Minimal Working Example

The conic optimization problem (11.6) , has three variables in a quadratic cone - first one is integer - and an affine expression in domain 0 (equality constraint).

$$\begin{array}{ll} \text{minimize} & 5.1\,x_0 \\ \text{subject to} & 6.2\,x_1 + 7.3\,x_2 - 8.4 \in \{0\} \\ & x \in \mathcal{Q}^3,\ x_0 \in \mathbb{Z}. \end{array} \tag{11.6}$$

Its formulation in the Conic Benchmark Format begins with the version of the CBF format used, to safeguard against later revisions.

```
VER
1
```

Next follows the problem structure, consisting of the objective sense, the number and domain of variables, the indices of integer variables, and the number and domain of scalar-valued affine expressions (i.e., the equality constraint).

```
OBJSENSE
MIN

VAR
3 1
Q 3

INT
1
0

CON
1 1
L= 1
```

Finally follows the problem data, consisting of the coefficients of the objective, the coefficients of the constraints, and the constant terms of the constraints. All data is specified on a sparse coordinate form.

```
OBJACOORD
1
0 5.1

ACOORD
2
0 1 6.2
0 2 7.3

BCOORD
1
0 -8.4
```

This concludes the example.

Mixing Linear, Second-order and Semidefinite Cones

The conic optimization problem (11.7), has a semidefinite cone, a quadratic cone over unordered subindices, and two equality constraints.

$$
\begin{array}{lrcl}
\text{minimize} & \left\langle \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}, X_1 \right\rangle + x_1 & & \\
\text{subject to} & \left\langle \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, X_1 \right\rangle + x_1 & = & 1.0\,, \\
& \left\langle \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, X_1 \right\rangle + x_0 + x_2 & = & 0.5\,, \\
& x_1 \geq \sqrt{x_0^2 + x_2^2}\,, & & \\
& X_1 \succeq \mathbf{0}\,. & &
\end{array}
\tag{11.7}
$$

The equality constraints are easily rewritten to the conic form, $(g_0, g_1) \in \{0\}^2$, by moving constants such that the right-hand-side becomes zero. The quadratic cone does not fit under the `VAR` keyword in this variable permutation. Instead, it takes a scalar constraint $(g_2, g_3, g_4) = (x_1, x_0, x_2) \in \mathcal{Q}^3$, with scalar variables constructed as $(x_0, x_1, x_2) \in \mathbb{R}^3$. Its formulation in the CBF format is reported in the following list

```
# File written using this version of the Conic Benchmark Format:
#     | Version 1.
VER
1

# The sense of the objective is:
#     | Minimize.
OBJSENSE
MIN

# One PSD variable of this size:
#     | Three times three.
PSDVAR
1
3

# Three scalar variables in this one conic domain:
#     | Three are free.
VAR
3 1
F 3

# Five scalar constraints with affine expressions in two conic domains:
#     | Two are fixed to zero.
#     | Three are in conic quadratic domain.
CON
5 2
L= 2
Q 3

# Five coordinates in F^{obj}_j coefficients:
#     | F^{obj}[0][0,0] = 2.0
#     | F^{obj}[0][1,0] = 1.0
#     | and more...
OBJFCOORD
5
0 0 0 2.0
0 1 0 1.0
0 1 1 2.0
```

```
0 2 1 1.0
0 2 2 2.0

# One coordinate in a^{obj}_j coefficients:
#     | a^{obj}[1] = 1.0
OBJACOORD
1
1 1.0

# Nine coordinates in F_ij coefficients:
#     | F[0,0][0,0] = 1.0
#     | F[0,0][1,1] = 1.0
#     | and more...
FCOORD
9
0 0 0 0 1.0
0 0 1 1 1.0
0 0 2 2 1.0
1 0 0 0 1.0
1 0 1 0 1.0
1 0 2 0 1.0
1 0 1 1 1.0
1 0 2 1 1.0
1 0 2 2 1.0

# Six coordinates in a_ij coefficients:
#     | a[0,1] = 1.0
#     | a[1,0] = 1.0
#     | and more...
ACOORD
6
0 1 1.0
1 0 1.0
1 2 1.0
2 1 1.0
3 0 1.0
4 2 1.0

# Two coordinates in b_i coefficients:
#     | b[0] = -1.0
#     | b[1] = -0.5
BCOORD
2
0 -1.0
1 -0.5
```

Mixing Semidefinite Variables and Linear Matrix Inequalities

The standard forms in semidefinite optimization are usually based either on semidefinite variables or linear matrix inequalities. In the CBF format, both forms are supported and can even be mixed as shown in.

$$
\begin{array}{lll}
\text{minimize} & \left\langle \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, X_1 \right\rangle + x_1 + x_2 + 1 & \\
\text{subject to} & \left\langle \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, X_1 \right\rangle - x_1 - x_2 & \geq \; 0.0\,, \\
 & x_1 \begin{bmatrix} 0 & 1 \\ 1 & 3 \end{bmatrix} + x_2 \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \succeq \; \mathbf{0}\,, \\
 & X_1 \succeq \mathbf{0}\,. &
\end{array} \tag{11.8}
$$

Its formulation in the CBF format is written in what follows

```
# File written using this version of the Conic Benchmark Format:
#     | Version 1.
VER
1

# The sense of the objective is:
#     | Minimize.
OBJSENSE
MIN

# One PSD variable of this size:
#     | Two times two.
PSDVAR
1
2

# Two scalar variables in this one conic domain:
#     | Two are free.
VAR
2 1
F 2

# One PSD constraint of this size:
#     | Two times two.
PSDCON
1
2

# One scalar constraint with an affine expression in this one conic domain:
#     | One is greater than or equal to zero.
CON
1 1
L+ 1

# Two coordinates in F^{obj}_j coefficients:
#     | F^{obj}[0][0,0] = 1.0
#     | F^{obj}[0][1,1] = 1.0
OBJFCOORD
2
0 0 0 1.0
0 1 1 1.0

# Two coordinates in a^{obj}_j coefficients:
#     | a^{obj}[0] = 1.0
#     | a^{obj}[1] = 1.0
OBJACOORD
2
0 1.0
1 1.0

# One coordinate in b^{obj} coefficient:
#     | b^{obj} = 1.0
OBJBCOORD
1.0

# One coordinate in F_ij coefficients:
#     | F[0,0][1,0] = 1.0
FCOORD
1
0 0 1 0 1.0

# Two coordinates in a_ij coefficients:
#     | a[0,0] = -1.0
```

```
#     | a[0,1] = -1.0
ACOORD
2
0 0 -1.0
0 1 -1.0

# Four coordinates in H_ij coefficients:
#     | H[0,0][1,0] = 1.0
#     | H[0,0][1,1] = 3.0
#     | and more...
HCOORD
4
0 0 1 0 1.0
0 0 1 1 3.0
0 1 0 0 3.0
0 1 1 0 1.0

# Two coordinates in D_i coefficients:
#     | D[0][0,0] = -1.0
#     | D[0][1,1] = -1.0
DCOORD
2
0 0 0 -1.0
0 1 1 -1.0
```

Optimization Over a Sequence of Objectives

The linear optimization problem (11.9), is defined for a sequence of objectives such that hotstarting from one to the next might be advantages.

$$
\begin{array}{lrcl}
\text{maximize}_k & g_k^{obj} & & \\
\text{subject to} & 50\,x_0 + 31 & \leq & 250\,, \\
 & 3\,x_0 - 2x_1 & \geq & -4\,, \\
 & x \in \mathbb{R}_+^2, & &
\end{array} \tag{11.9}
$$

given,

1. $g_0^{obj} = x_0 + 0.64x_1$.
2. $g_1^{obj} = 1.11x_0 + 0.76x_1$.
3. $g_2^{obj} = 1.11x_0 + 0.85x_1$.

Its formulation in the CBF format is reported in Listing 11.5.

Listing 11.5: Problem (11.9) in CBF format.

```
# File written using this version of the Conic Benchmark Format:
#     | Version 1.
VER
1

# The sense of the objective is:
#     | Maximize.
OBJSENSE
MAX

# Two scalar variables in this one conic domain:
#     | Two are nonnegative.
VAR
2 1
L+ 2
```

```
# Two scalar constraints with affine expressions in these two conic domains:
#     | One is in the nonpositive domain.
#     | One is in the nonnegative domain.
CON
2 2
L- 1
L+ 1

# Two coordinates in a^{obj}_j coefficients:
#     | a^{obj}[0] = 1.0
#     | a^{obj}[1] = 0.64
OBJACOORD
2
0 1.0
1 0.64

# Four coordinates in a_ij coefficients:
#     | a[0,0] = 50.0
#     | a[1,0] = 3.0
#     | and more...
ACOORD
4
0 0 50.0
1 0 3.0
0 1 31.0
1 1 -2.0

# Two coordinates in b_i coefficients:
#     | b[0] = -250.0
#     | b[1] = 4.0
BCOORD
2
0 -250.0
1 4.0

# New problem instance defined in terms of changes.
CHANGE

# Two coordinate changes in a^{obj}_j coefficients. Now it is:
#     | a^{obj}[0] = 1.11
#     | a^{obj}[1] = 0.76
OBJACOORD
2
0 1.11
1 0.76

# New problem instance defined in terms of changes.
CHANGE

# One coordinate change in a^{obj}_j coefficients. Now it is:
#     | a^{obj}[0] = 1.11
#     | a^{obj}[1] = 0.85
OBJACOORD
1
1 0.85
```

11.5 The XML (OSiL) Format

MOSEK can write data in the standard OSiL xml format. For a definition of the OSiL format please see http://www.optimizationservices.org/.

Only linear constraints (possibly with integer variables) are supported. By default output files with the extension `.xml` are written in the OSiL format.

The parameter *MSK_IPAR_WRITE_XML_MODE* controls if the linear coefficients in the A matrix are written in row or column order.

11.6 The Task Format

The Task format is **MOSEK**'s native binary format. It contains a complete image of a **MOSEK** task, i.e.

- Problem data: Linear, conic quadratic, semidefinite and quadratic data
- Problem item names: Variable names, constraints names, cone names etc.
- Parameter settings
- Solutions

There are a few things to be aware of:

- The task format *does not* support General Convex problems since these are defined by arbitrary user-defined functions.
- Status of a solution read from a file will *always* be unknown.

The format is based on the *TAR* (USTar) file format. This means that the individual pieces of data in a `.task` file can be examined by unpacking it as a *TAR* file. Please note that the inverse may not work: Creating a file using *TAR* will most probably not create a valid **MOSEK** Task file since the order of the entries is important.

11.7 The JSON Format

MOSEK provides the possibility to read/write problems in valid JSON format.

> JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

The official JSON website http://www.json.org provides plenty of information along with the format definition.

MOSEK defines two JSON-like formats:

- *jtask*
- *jsol*

Warning: Despite being text-based human-readable formats, *jtask* and *jsol* files will include no indentation and no new-lines, in order to keep the files as compact as possible. We therefore strongly advise to use JSON viewer tools to inspect *jtask* and *jsol* files.

11.7.1 *jtask* format

It stores a problem instance. The *jtask* format contains the same information as a *task format*.

Even though a *jtask* file is human-readable, we do not recommend users to create it by hand, but to rely on **MOSEK**.

11.7.2 *jsol* format

It stores a problem solution. The *jsol* format contains all solutions and information items.

11.7.3 A *jtask* example

In Listing 11.6 we present a file in the *jtask* format that corresponds to the sample problem from `lo1.lp`. The listing has been formatted for readability.

Listing 11.6: A formatted *jtask* file for the `lo1.lp` example.

```
{
    "$schema":"http://mosek.com/json/schema#",
    "Task/INFO":{
        "taskname":"lo1",
        "numvar":4,
        "numcon":3,
        "numcone":0,
        "numbarvar":0,
        "numanz":9,
        "numsymmat":0,
        "mosekver":[
            8,
            0,
            0,
            9
        ]
    },
    "Task/data":{
        "var":{
            "name":[
                "x1",
                "x2",
                "x3",
                "x4"
            ],
            "bk":[
                "lo",
                "ra",
                "lo",
                "lo"
            ],
            "bl":[
                0.0,
                0.0,
                0.0,
                0.0
            ],
            "bu":[
                1e+30,
                1e+1,
                1e+30,
                1e+30
            ],
```

```
            "type":[
                "cont",
                "cont",
                "cont",
                "cont"
            ]
        },
        "con":{
            "name":[
                "c1",
                "c2",
                "c3"
            ],
            "bk":[
                "fx",
                "lo",
                "up"
            ],
            "bl":[
                3e+1,
                1.5e+1,
                    -1e+30
            ],
            "bu":[
                3e+1,
                1e+30,
                2.5e+1
            ]
        },
        "objective":{
            "sense":"max",
            "name":"obj",
            "c":{
                "subj":[
                    0,
                    1,
                    2,
                    3
                ],
                "val":[
                    3e+0,
                    1e+0,
                    5e+0,
                    1e+0
                ]
            },
            "cfix":0.0
        },
        "A":{
            "subi":[
                0,
                0,
                0,
                1,
                1,
                1,
                1,
                2,
                2
            ],
            "subj":[
                0,
                1,
```

```
                2,
                0,
                1,
                2,
                3,
                1,
                3
            ],
            "val":[
                3e+0,
                1e+0,
                2e+0,
                2e+0,
                1e+0,
                3e+0,
                1e+0,
                2e+0,
                3e+0
            ]
        }
    },
    "Task/parameters":{
        "iparam":{
            "ANA_SOL_BASIS":"ON",
            "ANA_SOL_PRINT_VIOLATED":"OFF",
            "AUTO_SORT_A_BEFORE_OPT":"OFF",
            "AUTO_UPDATE_SOL_INFO":"OFF",
            "BASIS_SOLVE_USE_PLUS_ONE":"OFF",
            "BI_CLEAN_OPTIMIZER":"OPTIMIZER_FREE",
            "BI_IGNORE_MAX_ITER":"OFF",
            "BI_IGNORE_NUM_ERROR":"OFF",
            "BI_MAX_ITERATIONS":1000000,
            "CACHE_LICENSE":"ON",
            "CHECK_CONVEXITY":"CHECK_CONVEXITY_FULL",
            "COMPRESS_STATFILE":"ON",
            "CONCURRENT_NUM_OPTIMIZERS":2,
            "CONCURRENT_PRIORITY_DUAL_SIMPLEX":2,
            "CONCURRENT_PRIORITY_FREE_SIMPLEX":3,
            "CONCURRENT_PRIORITY_INTPNT":4,
            "CONCURRENT_PRIORITY_PRIMAL_SIMPLEX":1,
            "FEASREPAIR_OPTIMIZE":"FEASREPAIR_OPTIMIZE_NONE",
            "INFEAS_GENERIC_NAMES":"OFF",
            "INFEAS_PREFER_PRIMAL":"ON",
            "INFEAS_REPORT_AUTO":"OFF",
            "INFEAS_REPORT_LEVEL":1,
            "INTPNT_BASIS":"BI_ALWAYS",
            "INTPNT_DIFF_STEP":"ON",
            "INTPNT_FACTOR_DEBUG_LVL":0,
            "INTPNT_FACTOR_METHOD":0,
            "INTPNT_HOTSTART":"INTPNT_HOTSTART_NONE",
            "INTPNT_MAX_ITERATIONS":400,
            "INTPNT_MAX_NUM_COR":-1,
            "INTPNT_MAX_NUM_REFINEMENT_STEPS":-1,
            "INTPNT_OFF_COL_TRH":40,
            "INTPNT_ORDER_METHOD":"ORDER_METHOD_FREE",
            "INTPNT_REGULARIZATION_USE":"ON",
            "INTPNT_SCALING":"SCALING_FREE",
            "INTPNT_SOLVE_FORM":"SOLVE_FREE",
            "INTPNT_STARTING_POINT":"STARTING_POINT_FREE",
            "LIC_TRH_EXPIRY_WRN":7,
            "LICENSE_DEBUG":"OFF",
            "LICENSE_PAUSE_TIME":0,
            "LICENSE_SUPPRESS_EXPIRE_WRNS":"OFF",
```

```
            "LICENSE_WAIT":"OFF",
            "LOG":10,
            "LOG_ANA_PRO":1,
            "LOG_BI":4,
            "LOG_BI_FREQ":2500,
            "LOG_CHECK_CONVEXITY":0,
            "LOG_CONCURRENT":1,
            "LOG_CUT_SECOND_OPT":1,
            "LOG_EXPAND":0,
            "LOG_FACTOR":1,
            "LOG_FEAS_REPAIR":1,
            "LOG_FILE":1,
            "LOG_HEAD":1,
            "LOG_INFEAS_ANA":1,
            "LOG_INTPNT":4,
            "LOG_MIO":4,
            "LOG_MIO_FREQ":1000,
            "LOG_OPTIMIZER":1,
            "LOG_ORDER":1,
            "LOG_PRESOLVE":1,
            "LOG_RESPONSE":0,
            "LOG_SENSITIVITY":1,
            "LOG_SENSITIVITY_OPT":0,
            "LOG_SIM":4,
            "LOG_SIM_FREQ":1000,
            "LOG_SIM_MINOR":1,
            "LOG_STORAGE":1,
            "MAX_NUM_WARNINGS":10,
            "MIO_BRANCH_DIR":"BRANCH_DIR_FREE",
            "MIO_CONSTRUCT_SOL":"OFF",
            "MIO_CUT_CLIQUE":"ON",
            "MIO_CUT_CMIR":"ON",
            "MIO_CUT_GMI":"ON",
            "MIO_CUT_KNAPSACK_COVER":"OFF",
            "MIO_HEURISTIC_LEVEL":-1,
            "MIO_MAX_NUM_BRANCHES":-1,
            "MIO_MAX_NUM_RELAXS":-1,
            "MIO_MAX_NUM_SOLUTIONS":-1,
            "MIO_MODE":"MIO_MODE_SATISFIED",
            "MIO_MT_USER_CB":"ON",
            "MIO_NODE_OPTIMIZER":"OPTIMIZER_FREE",
            "MIO_NODE_SELECTION":"MIO_NODE_SELECTION_FREE",
            "MIO_PERSPECTIVE_REFORMULATE":"ON",
            "MIO_PROBING_LEVEL":-1,
            "MIO_RINS_MAX_NODES":-1,
            "MIO_ROOT_OPTIMIZER":"OPTIMIZER_FREE",
            "MIO_ROOT_REPEAT_PRESOLVE_LEVEL":-1,
            "MT_SPINCOUNT":0,
            "NUM_THREADS":0,
            "OPF_MAX_TERMS_PER_LINE":5,
            "OPF_WRITE_HEADER":"ON",
            "OPF_WRITE_HINTS":"ON",
            "OPF_WRITE_PARAMETERS":"OFF",
            "OPF_WRITE_PROBLEM":"ON",
            "OPF_WRITE_SOL_BAS":"ON",
            "OPF_WRITE_SOL_ITG":"ON",
            "OPF_WRITE_SOL_ITR":"ON",
            "OPF_WRITE_SOLUTIONS":"OFF",
            "OPTIMIZER":"OPTIMIZER_FREE",
            "PARAM_READ_CASE_NAME":"ON",
            "PARAM_READ_IGN_ERROR":"OFF",
            "PRESOLVE_ELIMINATOR_MAX_FILL":-1,
            "PRESOLVE_ELIMINATOR_MAX_NUM_TRIES":-1,
```

```
            "PRESOLVE_LEVEL":-1,
            "PRESOLVE_LINDEP_ABS_WORK_TRH":100,
            "PRESOLVE_LINDEP_REL_WORK_TRH":100,
            "PRESOLVE_LINDEP_USE":"ON",
            "PRESOLVE_MAX_NUM_REDUCTIONS":-1,
            "PRESOLVE_USE":"PRESOLVE_MODE_FREE",
            "PRIMAL_REPAIR_OPTIMIZER":"OPTIMIZER_FREE",
            "QO_SEPARABLE_REFORMULATION":"OFF",
            "READ_DATA_COMPRESSED":"COMPRESS_FREE",
            "READ_DATA_FORMAT":"DATA_FORMAT_EXTENSION",
            "READ_DEBUG":"OFF",
            "READ_KEEP_FREE_CON":"OFF",
            "READ_LP_DROP_NEW_VARS_IN_BOU":"OFF",
            "READ_LP_QUOTED_NAMES":"ON",
            "READ_MPS_FORMAT":"MPS_FORMAT_FREE",
            "READ_MPS_WIDTH":1024,
            "READ_TASK_IGNORE_PARAM":"OFF",
            "SENSITIVITY_ALL":"OFF",
            "SENSITIVITY_OPTIMIZER":"OPTIMIZER_FREE_SIMPLEX",
            "SENSITIVITY_TYPE":"SENSITIVITY_TYPE_BASIS",
            "SIM_BASIS_FACTOR_USE":"ON",
            "SIM_DEGEN":"SIM_DEGEN_FREE",
            "SIM_DUAL_CRASH":90,
            "SIM_DUAL_PHASEONE_METHOD":0,
            "SIM_DUAL_RESTRICT_SELECTION":50,
            "SIM_DUAL_SELECTION":"SIM_SELECTION_FREE",
            "SIM_EXPLOIT_DUPVEC":"SIM_EXPLOIT_DUPVEC_OFF",
            "SIM_HOTSTART":"SIM_HOTSTART_FREE",
            "SIM_HOTSTART_LU":"ON",
            "SIM_INTEGER":0,
            "SIM_MAX_ITERATIONS":10000000,
            "SIM_MAX_NUM_SETBACKS":250,
            "SIM_NON_SINGULAR":"ON",
            "SIM_PRIMAL_CRASH":90,
            "SIM_PRIMAL_PHASEONE_METHOD":0,
            "SIM_PRIMAL_RESTRICT_SELECTION":50,
            "SIM_PRIMAL_SELECTION":"SIM_SELECTION_FREE",
            "SIM_REFACTOR_FREQ":0,
            "SIM_REFORMULATION":"SIM_REFORMULATION_OFF",
            "SIM_SAVE_LU":"OFF",
            "SIM_SCALING":"SCALING_FREE",
            "SIM_SCALING_METHOD":"SCALING_METHOD_POW2",
            "SIM_SOLVE_FORM":"SOLVE_FREE",
            "SIM_STABILITY_PRIORITY":50,
            "SIM_SWITCH_OPTIMIZER":"OFF",
            "SOL_FILTER_KEEP_BASIC":"OFF",
            "SOL_FILTER_KEEP_RANGED":"OFF",
            "SOL_READ_NAME_WIDTH":-1,
            "SOL_READ_WIDTH":1024,
            "SOLUTION_CALLBACK":"OFF",
            "TIMING_LEVEL":1,
            "WRITE_BAS_CONSTRAINTS":"ON",
            "WRITE_BAS_HEAD":"ON",
            "WRITE_BAS_VARIABLES":"ON",
            "WRITE_DATA_COMPRESSED":0,
            "WRITE_DATA_FORMAT":"DATA_FORMAT_EXTENSION",
            "WRITE_DATA_PARAM":"OFF",
            "WRITE_FREE_CON":"OFF",
            "WRITE_GENERIC_NAMES":"OFF",
            "WRITE_GENERIC_NAMES_IO":1,
            "WRITE_IGNORE_INCOMPATIBLE_CONIC_ITEMS":"OFF",
            "WRITE_IGNORE_INCOMPATIBLE_ITEMS":"OFF",
            "WRITE_IGNORE_INCOMPATIBLE_NL_ITEMS":"OFF",
```

```
        "WRITE_IGNORE_INCOMPATIBLE_PSD_ITEMS":"OFF",
        "WRITE_INT_CONSTRAINTS":"ON",
        "WRITE_INT_HEAD":"ON",
        "WRITE_INT_VARIABLES":"ON",
        "WRITE_LP_FULL_OBJ":"ON",
        "WRITE_LP_LINE_WIDTH":80,
        "WRITE_LP_QUOTED_NAMES":"ON",
        "WRITE_LP_STRICT_FORMAT":"OFF",
        "WRITE_LP_TERMS_PER_LINE":10,
        "WRITE_MPS_FORMAT":"MPS_FORMAT_FREE",
        "WRITE_MPS_INT":"ON",
        "WRITE_PRECISION":15,
        "WRITE_SOL_BARVARIABLES":"ON",
        "WRITE_SOL_CONSTRAINTS":"ON",
        "WRITE_SOL_HEAD":"ON",
        "WRITE_SOL_IGNORE_INVALID_NAMES":"OFF",
        "WRITE_SOL_VARIABLES":"ON",
        "WRITE_TASK_INC_SOL":"ON",
        "WRITE_XML_MODE":"WRITE_XML_MODE_ROW"
    },
    "dparam":{
        "ANA_SOL_INFEAS_TOL":1e-6,
        "BASIS_REL_TOL_S":1e-12,
        "BASIS_TOL_S":1e-6,
        "BASIS_TOL_X":1e-6,
        "CHECK_CONVEXITY_REL_TOL":1e-10,
        "DATA_TOL_AIJ":1e-12,
        "DATA_TOL_AIJ_HUGE":1e+20,
        "DATA_TOL_AIJ_LARGE":1e+10,
        "DATA_TOL_BOUND_INF":1e+16,
        "DATA_TOL_BOUND_WRN":1e+8,
        "DATA_TOL_C_HUGE":1e+16,
        "DATA_TOL_CJ_LARGE":1e+8,
        "DATA_TOL_QIJ":1e-16,
        "DATA_TOL_X":1e-8,
        "FEASREPAIR_TOL":1e-10,
        "INTPNT_CO_TOL_DFEAS":1e-8,
        "INTPNT_CO_TOL_INFEAS":1e-10,
        "INTPNT_CO_TOL_MU_RED":1e-8,
        "INTPNT_CO_TOL_NEAR_REL":1e+3,
        "INTPNT_CO_TOL_PFEAS":1e-8,
        "INTPNT_CO_TOL_REL_GAP":1e-7,
        "INTPNT_NL_MERIT_BAL":1e-4,
        "INTPNT_NL_TOL_DFEAS":1e-8,
        "INTPNT_NL_TOL_MU_RED":1e-12,
        "INTPNT_NL_TOL_NEAR_REL":1e+3,
        "INTPNT_NL_TOL_PFEAS":1e-8,
        "INTPNT_NL_TOL_REL_GAP":1e-6,
        "INTPNT_NL_TOL_REL_STEP":9.95e-1,
        "INTPNT_QO_TOL_DFEAS":1e-8,
        "INTPNT_QO_TOL_INFEAS":1e-10,
        "INTPNT_QO_TOL_MU_RED":1e-8,
        "INTPNT_QO_TOL_NEAR_REL":1e+3,
        "INTPNT_QO_TOL_PFEAS":1e-8,
        "INTPNT_QO_TOL_REL_GAP":1e-8,
        "INTPNT_TOL_DFEAS":1e-8,
        "INTPNT_TOL_DSAFE":1e+0,
        "INTPNT_TOL_INFEAS":1e-10,
        "INTPNT_TOL_MU_RED":1e-16,
        "INTPNT_TOL_PATH":1e-8,
        "INTPNT_TOL_PFEAS":1e-8,
        "INTPNT_TOL_PSAFE":1e+0,
        "INTPNT_TOL_REL_GAP":1e-8,
```

```
            "INTPNT_TOL_REL_STEP":9.999e-1,
            "INTPNT_TOL_STEP_SIZE":1e-6,
            "LOWER_OBJ_CUT":-1e+30,
            "LOWER_OBJ_CUT_FINITE_TRH":-5e+29,
            "MIO_DISABLE_TERM_TIME":-1e+0,
            "MIO_MAX_TIME":-1e+0,
            "MIO_MAX_TIME_APRX_OPT":6e+1,
            "MIO_NEAR_TOL_ABS_GAP":0.0,
            "MIO_NEAR_TOL_REL_GAP":1e-3,
            "MIO_REL_GAP_CONST":1e-10,
            "MIO_TOL_ABS_GAP":0.0,
            "MIO_TOL_ABS_RELAX_INT":1e-5,
            "MIO_TOL_FEAS":1e-6,
            "MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT":0.0,
            "MIO_TOL_REL_GAP":1e-4,
            "MIO_TOL_X":1e-6,
            "OPTIMIZER_MAX_TIME":-1e+0,
            "PRESOLVE_TOL_ABS_LINDEP":1e-6,
            "PRESOLVE_TOL_AIJ":1e-12,
            "PRESOLVE_TOL_REL_LINDEP":1e-10,
            "PRESOLVE_TOL_S":1e-8,
            "PRESOLVE_TOL_X":1e-8,
            "QCQO_REFORMULATE_REL_DROP_TOL":1e-15,
            "SEMIDEFINITE_TOL_APPROX":1e-10,
            "SIM_LU_TOL_REL_PIV":1e-2,
            "SIMPLEX_ABS_TOL_PIV":1e-7,
            "UPPER_OBJ_CUT":1e+30,
            "UPPER_OBJ_CUT_FINITE_TRH":5e+29
        },
        "sparam":{
            "BAS_SOL_FILE_NAME":"",
            "DATA_FILE_NAME":"examples/tools/data/lo1.mps",
            "DEBUG_FILE_NAME":"",
            "INT_SOL_FILE_NAME":"",
            "ITR_SOL_FILE_NAME":"",
            "MIO_DEBUG_STRING":"",
            "PARAM_COMMENT_SIGN":"%%",
            "PARAM_READ_FILE_NAME":"",
            "PARAM_WRITE_FILE_NAME":"",
            "READ_MPS_BOU_NAME":"",
            "READ_MPS_OBJ_NAME":"",
            "READ_MPS_RAN_NAME":"",
            "READ_MPS_RHS_NAME":"",
            "SENSITIVITY_FILE_NAME":"",
            "SENSITIVITY_RES_FILE_NAME":"",
            "SOL_FILTER_XC_LOW":"",
            "SOL_FILTER_XC_UPR":"",
            "SOL_FILTER_XX_LOW":"",
            "SOL_FILTER_XX_UPR":"",
            "STAT_FILE_NAME":"",
            "STAT_KEY":"",
            "STAT_NAME":"",
            "WRITE_LP_GEN_VAR_NAME":"XMSKGEN"
        }
    }
}
```


11.8 The Solution File Format

MOSEK provides several solution files depending on the problem type and the optimizer used:

- *basis solution file* (extension .bas) if the problem is optimized using the simplex optimizer or basis identification is performed,
- *interior solution file* (extension .sol) if a problem is optimized using the interior-point optimizer and no basis identification is required,
- *integer solution file* (extension .int) if the problem contains integer constrained variables.

All solution files have the format:

```
NAME                : <problem name>
PROBLEM STATUS      : <status of the problem>
SOLUTION STATUS     : <status of the solution>
OBJECTIVE NAME      : <name of the objective function>
PRIMAL OBJECTIVE    : <primal objective value corresponding to the solution>
DUAL OBJECTIVE      : <dual objective value corresponding to the solution>
CONSTRAINTS
INDEX   NAME     AT ACTIVITY    LOWER LIMIT   UPPER LIMIT   DUAL LOWER   DUAL UPPER
?       <name>   ?? <a value>   <a value>     <a value>     <a value>    <a value>
VARIABLES
INDEX   NAME     AT ACTIVITY    LOWER LIMIT   UPPER LIMIT   DUAL LOWER   DUAL UPPER    CONIC␣
↪DUAL
?       <name>   ?? <a value>   <a value>     <a value>     <a value>    <a value>     <a value>
```

In the example the fields ? and <> will be filled with problem and solution specific information. As can be observed a solution report consists of three sections, i.e.

- HEADER In this section, first the name of the problem is listed and afterwards the problem and solution status are shown. Next the primal and dual objective values are displayed.
- CONSTRAINTS For each constraint i of the form

$$l_i^c \leq \sum_{j=1}^{n} a_{ij}x_j \leq u_i^c, \tag{11.10}$$

  the following information is listed:

  - INDEX: A sequential index assigned to the constraint by **MOSEK**
  - NAME: The name of the constraint assigned by the user.
  - AT: The status of the constraint. In Table 11.4 the possible values of the status keys and their interpretation are shown.

Table 11.4: Status keys.

Status key	Interpretation
UN	Unknown status
BS	Is basic
SB	Is superbasic
LL	Is at the lower limit (bound)
UL	Is at the upper limit (bound)
EQ	Lower limit is identical to upper limit
**	Is infeasible i.e. the lower limit is greater than the upper limit.

  - ACTIVITY: the quantity $\sum_{j=1}^{n} a_{ij}x_j^*$, where x^* is the value of the primal solution.
  - LOWER LIMIT: the quantity l_i^c (see (11.10).)
  - UPPER LIMIT: the quantity u_i^c (see (11.10).)
  - DUAL LOWER: the dual multiplier corresponding to the lower limit on the constraint.
  - DUAL UPPER: the dual multiplier corresponding to the upper limit on the constraint.

- VARIABLES The last section of the solution report lists information about the variables. This information has a similar interpretation as for the constraints. However, the column with the header CONIC DUAL is included for problems having one or more conic constraints. This column shows the dual variables corresponding to the conic constraints.

Example: lo1.sol

In Listing 11.7 we show the solution file for the lo1.opf problem.

Listing 11.7: An example of .sol file.

```
NAME                :
PROBLEM STATUS      : PRIMAL_AND_DUAL_FEASIBLE
SOLUTION STATUS     : OPTIMAL
OBJECTIVE NAME      : obj
PRIMAL OBJECTIVE    : 8.33333333e+01
DUAL OBJECTIVE      : 8.33333332e+01

CONSTRAINTS
INDEX      NAME           AT ACTIVITY                 LOWER LIMIT        UPPER LIMIT        ␣
↪DUAL LOWER                DUAL UPPER
0          c1             EQ 3.00000000000000e+01     3.00000000e+01     3.00000000e+01     -0.
↪00000000000000e+00     -2.49999999741654e+00
1          c2             SB 5.33333333049188e+01     1.50000000e+01     NONE               2.
↪09157603759397e-10      -0.00000000000000e+00
2          c3             UL 2.49999999842049e+01     NONE               2.50000000e+01     -0.
↪00000000000000e+00     -3.33333332895110e-01

VARIABLES
INDEX      NAME           AT ACTIVITY                 LOWER LIMIT        UPPER LIMIT        ␣
↪DUAL LOWER                DUAL UPPER
0          x1             LL 1.67020427073508e-09     0.00000000e+00     NONE               -4.
↪49999999528055e+00     -0.00000000000000e+00
1          x2             LL 2.93510446280504e-09     0.00000000e+00     1.00000000e+01     -2.
↪16666666494916e+00     6.20863861687316e-10
2          x3             SB 1.49999999899425e+01     0.00000000e+00     NONE               -8.
↪79123177454657e-10     -0.00000000000000e+00
3          x4             SB 8.33333332273116e+00     0.00000000e+00     NONE               -1.
↪69795978899185e-09     -0.00000000000000e+00
```

CHAPTER

TWELVE

INTERFACE CHANGES

The section show interface-specific changes to the **MOSEK** Rmosek Package in version 8. See the release notes for general changes and new features of the **MOSEK** Optimization Suite.

12.1 Parameters

Added

- *MSK_DPAR_DATA_SYM_MAT_TOL*
- *MSK_DPAR_DATA_SYM_MAT_TOL_HUGE*
- *MSK_DPAR_DATA_SYM_MAT_TOL_LARGE*
- *MSK_DPAR_INTPNT_QO_TOL_DFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_INFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_MU_RED*
- *MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_QO_TOL_PFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_REL_GAP*
- *MSK_DPAR_SEMIDEFINITE_TOL_APPROX*
- *MSK_IPAR_INTPNT_MULTI_THREAD*
- *MSK_IPAR_LICENSE_TRH_EXPIRY_WRN*
- *MSK_IPAR_LOG_ANA_PRO*
- *MSK_IPAR_MIO_CUT_CLIQUE*
- *MSK_IPAR_MIO_CUT_GMI*
- *MSK_IPAR_MIO_CUT_IMPLIED_BOUND*
- *MSK_IPAR_MIO_CUT_KNAPSACK_COVER*
- *MSK_IPAR_MIO_CUT_SELECTION_LEVEL*
- *MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE*
- *MSK_IPAR_MIO_ROOT_REPEAT_PRESOLVE_LEVEL*
- *MSK_IPAR_MIO_VB_DETECTION_LEVEL*
- *MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_FILL*

Removed

- MSK_DPAR_FEASREPAIR_TOL
- MSK_DPAR_MIO_HEURISTIC_TIME
- MSK_DPAR_MIO_MAX_TIME_APRX_OPT
- MSK_DPAR_MIO_REL_ADD_CUT_LIMITED
- MSK_DPAR_MIO_TOL_MAX_CUT_FRAC_RHS
- MSK_DPAR_MIO_TOL_MIN_CUT_FRAC_RHS
- MSK_DPAR_MIO_TOL_REL_RELAX_INT
- MSK_DPAR_MIO_TOL_X
- MSK_DPAR_NONCONVEX_TOL_FEAS
- MSK_DPAR_NONCONVEX_TOL_OPT
- MSK_IPAR_ALLOC_ADD_QNZ
- MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS
- MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX
- MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX
- MSK_IPAR_CONCURRENT_PRIORITY_INTPNT
- MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX
- MSK_IPAR_FEASREPAIR_OPTIMIZE
- MSK_IPAR_INTPNT_FACTOR_DEBUG_LVL
- MSK_IPAR_INTPNT_FACTOR_METHOD
- MSK_IPAR_LIC_TRH_EXPIRY_WRN
- MSK_IPAR_LOG_CONCURRENT
- MSK_IPAR_LOG_NONCONVEX
- MSK_IPAR_LOG_PARAM
- MSK_IPAR_LOG_SIM_NETWORK_FREQ
- MSK_IPAR_MIO_BRANCH_PRIORITIES_USE
- MSK_IPAR_MIO_CONT_SOL
- MSK_IPAR_MIO_CUT_CG
- MSK_IPAR_MIO_CUT_LEVEL_ROOT
- MSK_IPAR_MIO_CUT_LEVEL_TREE
- MSK_IPAR_MIO_FEASPUMP_LEVEL
- MSK_IPAR_MIO_HOTSTART
- MSK_IPAR_MIO_KEEP_BASIS
- MSK_IPAR_MIO_LOCAL_BRANCH_NUMBER
- MSK_IPAR_MIO_OPTIMIZER_MODE
- MSK_IPAR_MIO_PRESOLVE_AGGREGATE
- MSK_IPAR_MIO_PRESOLVE_PROBING
- MSK_IPAR_MIO_PRESOLVE_USE

- MSK_IPAR_MIO_STRONG_BRANCH
- MSK_IPAR_MIO_USE_MULTITHREADED_OPTIMIZER
- MSK_IPAR_NONCONVEX_MAX_ITERATIONS
- MSK_IPAR_PRESOLVE_ELIM_FILL
- MSK_IPAR_PRESOLVE_ELIMINATOR_USE
- MSK_IPAR_QO_SEPARABLE_REFORMULATION
- MSK_IPAR_READ_ANZ
- MSK_IPAR_READ_CON
- MSK_IPAR_READ_CONE
- MSK_IPAR_READ_MPS_KEEP_INT
- MSK_IPAR_READ_MPS_OBJ_SENSE
- MSK_IPAR_READ_MPS_RELAX
- MSK_IPAR_READ_QNZ
- MSK_IPAR_READ_VAR
- MSK_IPAR_WARNING_LEVEL
- MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_CONIC_ITEMS
- MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_NL_ITEMS
- MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_PSD_ITEMS
- MSK_SPAR_FEASREPAIR_NAME_PREFIX
- MSK_SPAR_FEASREPAIR_NAME_SEPARATOR
- MSK_SPAR_FEASREPAIR_NAME_WSUMVIOL

12.2 Constants

Added

- *MSK_BRANCH_DIR_FAR*
- *MSK_BRANCH_DIR_GUIDED*
- *MSK_BRANCH_DIR_NEAR*
- *MSK_BRANCH_DIR_PSEUDOCOST*
- *MSK_BRANCH_DIR_ROOT_LP*
- *MSK_CALLBACK_BEGIN_ROOT_CUTGEN*
- *MSK_CALLBACK_BEGIN_TO_CONIC*
- *MSK_CALLBACK_END_ROOT_CUTGEN*
- *MSK_CALLBACK_END_TO_CONIC*
- *MSK_CALLBACK_IM_ROOT_CUTGEN*
- *MSK_CALLBACK_SOLVING_REMOTE*
- *MSK_DATA_FORMAT_JSON_TASK*
- *MSK_DINF_MIO_CLIQUE_SEPARATION_TIME*

- *MSK_DINF_MIO_CMIR_SEPARATION_TIME*
- *MSK_DINF_MIO_GMI_SEPARATION_TIME*
- *MSK_DINF_MIO_IMPLIED_BOUND_TIME*
- *MSK_DINF_MIO_KNAPSACK_COVER_SEPARATION_TIME*
- *MSK_DINF_QCQO_REFORMULATE_MAX_PERTURBATION*
- *MSK_DINF_QCQO_REFORMULATE_WORST_CHOLESKY_COLUMN_SCALING*
- *MSK_DINF_QCQO_REFORMULATE_WORST_CHOLESKY_DIAG_SCALING*
- *MSK_DINF_SOL_BAS_NRM_BARX*
- *MSK_DINF_SOL_BAS_NRM_SLC*
- *MSK_DINF_SOL_BAS_NRM_SLX*
- *MSK_DINF_SOL_BAS_NRM_SUC*
- *MSK_DINF_SOL_BAS_NRM_SUX*
- *MSK_DINF_SOL_BAS_NRM_XC*
- *MSK_DINF_SOL_BAS_NRM_XX*
- *MSK_DINF_SOL_BAS_NRM_Y*
- *MSK_DINF_SOL_ITG_NRM_BARX*
- *MSK_DINF_SOL_ITG_NRM_XC*
- *MSK_DINF_SOL_ITG_NRM_XX*
- *MSK_DINF_SOL_ITR_NRM_BARS*
- *MSK_DINF_SOL_ITR_NRM_BARX*
- *MSK_DINF_SOL_ITR_NRM_SLC*
- *MSK_DINF_SOL_ITR_NRM_SLX*
- *MSK_DINF_SOL_ITR_NRM_SNX*
- *MSK_DINF_SOL_ITR_NRM_SUC*
- *MSK_DINF_SOL_ITR_NRM_SUX*
- *MSK_DINF_SOL_ITR_NRM_XC*
- *MSK_DINF_SOL_ITR_NRM_XX*
- *MSK_DINF_SOL_ITR_NRM_Y*
- *MSK_DINF_TO_CONIC_TIME*
- *MSK_IINF_MIO_ABSGAP_SATISFIED*
- *MSK_IINF_MIO_CLIQUE_TABLE_SIZE*
- *MSK_IINF_MIO_NEAR_ABSGAP_SATISFIED*
- *MSK_IINF_MIO_NEAR_RELGAP_SATISFIED*
- *MSK_IINF_MIO_NODE_DEPTH*
- *MSK_IINF_MIO_NUM_CMIR_CUTS*
- *MSK_IINF_MIO_NUM_IMPLIED_BOUND_CUTS*
- *MSK_IINF_MIO_NUM_KNAPSACK_COVER_CUTS*
- *MSK_IINF_MIO_NUM_REPEATED_PRESOLVE*
- *MSK_IINF_MIO_PRESOLVED_NUMBIN*

- *MSK_IINF_MIO_PRESOLVED_NUMCON*
- *MSK_IINF_MIO_PRESOLVED_NUMCONT*
- *MSK_IINF_MIO_PRESOLVED_NUMINT*
- *MSK_IINF_MIO_PRESOLVED_NUMVAR*
- *MSK_IINF_MIO_RELGAP_SATISFIED*
- *MSK_LIINF_MIO_PRESOLVED_ANZ*
- *MSK_LIINF_MIO_SIM_MAXITER_SETBACKS*
- *MSK_MPS_FORMAT_CPLEX*
- *MSK_SOL_STA_DUAL_ILLPOSED_CER*
- *MSK_SOL_STA_PRIM_ILLPOSED_CER*

Changed

- *MSK_SOL_STA_INTEGER_OPTIMAL*
- *MSK_SOL_STA_NEAR_INTEGER_OPTIMAL*

Removed

- MSK_CALLBACKCODE_BEGIN_CONCURRENT
- MSK_CALLBACKCODE_BEGIN_NETWORK_DUAL_SIMPLEX
- MSK_CALLBACKCODE_BEGIN_NETWORK_PRIMAL_SIMPLEX
- MSK_CALLBACKCODE_BEGIN_NETWORK_SIMPLEX
- MSK_CALLBACKCODE_BEGIN_NONCONVEX
- MSK_CALLBACKCODE_BEGIN_SIMPLEX_NETWORK_DETECT
- MSK_CALLBACKCODE_END_CONCURRENT
- MSK_CALLBACKCODE_END_NETWORK_DUAL_SIMPLEX
- MSK_CALLBACKCODE_END_NETWORK_PRIMAL_SIMPLEX
- MSK_CALLBACKCODE_END_NETWORK_SIMPLEX
- MSK_CALLBACKCODE_END_NONCONVEX
- MSK_CALLBACKCODE_END_SIMPLEX_NETWORK_DETECT
- MSK_CALLBACKCODE_IM_MIO_PRESOLVE
- MSK_CALLBACKCODE_IM_NETWORK_DUAL_SIMPLEX
- MSK_CALLBACKCODE_IM_NETWORK_PRIMAL_SIMPLEX
- MSK_CALLBACKCODE_IM_NONCONVEX
- MSK_CALLBACKCODE_NONCOVEX
- MSK_CALLBACKCODE_UPDATE_NETWORK_DUAL_SIMPLEX
- MSK_CALLBACKCODE_UPDATE_NETWORK_PRIMAL_SIMPLEX
- MSK_CALLBACKCODE_UPDATE_NONCONVEX
- MSK_DINFITEM_CONCURRENT_TIME
- MSK_DINFITEM_MIO_CG_SEPERATION_TIME

- MSK_DINFITEM_MIO_CMIR_SEPERATION_TIME
- MSK_DINFITEM_SIM_NETWORK_DUAL_TIME
- MSK_DINFITEM_SIM_NETWORK_PRIMAL_TIME
- MSK_DINFITEM_SIM_NETWORK_TIME
- MSK_FEATURE_PTOM
- MSK_FEATURE_PTOX
- MSK_IINFITEM_CONCURRENT_FASTEST_OPTIMIZER
- MSK_IINFITEM_MIO_NUM_BASIS_CUTS
- MSK_IINFITEM_MIO_NUM_CARDGUB_CUTS
- MSK_IINFITEM_MIO_NUM_COEF_REDC_CUTS
- MSK_IINFITEM_MIO_NUM_CONTRA_CUTS
- MSK_IINFITEM_MIO_NUM_DISAGG_CUTS
- MSK_IINFITEM_MIO_NUM_FLOW_COVER_CUTS
- MSK_IINFITEM_MIO_NUM_GCD_CUTS
- MSK_IINFITEM_MIO_NUM_GUB_COVER_CUTS
- MSK_IINFITEM_MIO_NUM_KNAPSUR_COVER_CUTS
- MSK_IINFITEM_MIO_NUM_LATTICE_CUTS
- MSK_IINFITEM_MIO_NUM_LIFT_CUTS
- MSK_IINFITEM_MIO_NUM_OBJ_CUTS
- MSK_IINFITEM_MIO_NUM_PLAN_LOC_CUTS
- MSK_IINFITEM_SIM_NETWORK_DUAL_DEG_ITER
- MSK_IINFITEM_SIM_NETWORK_DUAL_HOTSTART
- MSK_IINFITEM_SIM_NETWORK_DUAL_HOTSTART_LU
- MSK_IINFITEM_SIM_NETWORK_DUAL_INF_ITER
- MSK_IINFITEM_SIM_NETWORK_DUAL_ITER
- MSK_IINFITEM_SIM_NETWORK_PRIMAL_DEG_ITER
- MSK_IINFITEM_SIM_NETWORK_PRIMAL_HOTSTART
- MSK_IINFITEM_SIM_NETWORK_PRIMAL_HOTSTART_LU
- MSK_IINFITEM_SIM_NETWORK_PRIMAL_INF_ITER
- MSK_IINFITEM_SIM_NETWORK_PRIMAL_ITER
- MSK_IINFITEM_SOL_INT_PROSTA
- MSK_IINFITEM_SOL_INT_SOLSTA
- MSK_IINFITEM_STO_NUM_A_CACHE_FLUSHES
- MSK_IINFITEM_STO_NUM_A_TRANSPOSES
- MSK_MIOMODE_LAZY
- MSK_OPTIMIZERTYPE_CONCURRENT
- MSK_OPTIMIZERTYPE_MIXED_INT_CONIC
- MSK_OPTIMIZERTYPE_NETWORK_PRIMAL_SIMPLEX
- MSK_OPTIMIZERTYPE_NONCONVEX

- MSK_OPTIMIZERTYPE_PRIMAL_DUAL_SIMPLEX

12.3 Response Codes

Added

- *MSK_RES_ERR_DUPLICATE_AIJ (1385)*
- *MSK_RES_ERR_JSON_DATA (1179)*
- *MSK_RES_ERR_JSON_FORMAT (1178)*
- *MSK_RES_ERR_JSON_MISSING_DATA (1180)*
- *MSK_RES_ERR_JSON_NUMBER_OVERFLOW (1177)*
- *MSK_RES_ERR_JSON_STRING (1176)*
- *MSK_RES_ERR_JSON_SYNTAX (1175)*
- *MSK_RES_ERR_LAU_INVALID_LOWER_TRIANGULAR_MATRIX (7002)*
- *MSK_RES_ERR_LAU_INVALID_SPARSE_SYMMETRIC_MATRIX (7019)*
- *MSK_RES_ERR_LAU_NOT_POSITIVE_DEFINITE (7001)*
- *MSK_RES_ERR_MIXED_CONIC_AND_NL (1501)*
- *MSK_RES_ERR_SERVER_CONNECT (8000)*
- *MSK_RES_ERR_SERVER_PROTOCOL (8001)*
- *MSK_RES_ERR_SERVER_STATUS (8002)*
- *MSK_RES_ERR_SERVER_TOKEN (8003)*
- *MSK_RES_ERR_SYM_MAT_HUGE (1482)*
- *MSK_RES_ERR_SYM_MAT_INVALID (1480)*
- *MSK_RES_ERR_TASK_WRITE (2562)*
- *MSK_RES_ERR_TOCONIC_CONSTR_NOT_CONIC (7153)*
- *MSK_RES_ERR_TOCONIC_CONSTR_Q_NOT_PSD (7150)*
- *MSK_RES_ERR_TOCONIC_CONSTRAINT_FX (7151)*
- *MSK_RES_ERR_TOCONIC_CONSTRAINT_RA (7152)*
- *MSK_RES_ERR_TOCONIC_OBJECTIVE_NOT_PSD (7155)*
- *MSK_RES_WRN_SYM_MAT_LARGE (960)*

Removed

- MSK_RES_ERR_AD_INVALID_OPERAND
- MSK_RES_ERR_AD_INVALID_OPERATOR
- MSK_RES_ERR_AD_MISSING_OPERAND
- MSK_RES_ERR_AD_MISSING_RETURN
- MSK_RES_ERR_CONCURRENT_OPTIMIZER
- MSK_RES_ERR_INV_CONIC_PROBLEM
- MSK_RES_ERR_INVALID_BRANCH_DIRECTION

- MSK_RES_ERR_INVALID_BRANCH_PRIORITY
- MSK_RES_ERR_INVALID_NETWORK_PROBLEM
- MSK_RES_ERR_MBT_INCOMPATIBLE
- MSK_RES_ERR_MBT_INVALID
- MSK_RES_ERR_MIXED_PROBLEM
- MSK_RES_ERR_NO_DUAL_INFO_FOR_ITG_SOL
- MSK_RES_ERR_ORD_INVALID
- MSK_RES_ERR_ORD_INVALID_BRANCH_DIR
- MSK_RES_ERR_TOCONIC_CONVERSION_FAIL
- MSK_RES_ERR_TOO_MANY_CONCURRENT_TASKS
- MSK_RES_WRN_TOO_MANY_THREADS_CONCURRENT

[AA95] E. D. Andersen and K. D. Andersen. Presolving in linear programming. *Math. Programming*, 71(2):221–245, 1995.

[AGMX96] E. D. Andersen, J. Gondzio, Cs. Mészáros, and X. Xu. Implementation of interior point methods for large scale linear programming. In T. Terlaky, editor, *Interior-point methods of mathematical programming*, pages 189–252. Kluwer Academic Publishers, 1996.

[ART03] E. D. Andersen, C. Roos, and T. Terlaky. On implementing a primal-dual interior-point method for conic quadratic optimization. *Math. Programming*, February 2003.

[AY96] E. D. Andersen and Y. Ye. Combining interior-point and pivoting algorithms. *Management Sci.*, 42(12):1719–1731, December 1996.

[AY98] E. D. Andersen and Y. Ye. A computational study of the homogeneous algorithm for large-scale convex optimization. *Computational Optimization and Applications*, 10:243–269, 1998.

[AY99] E. D. Andersen and Y. Ye. On a homogeneous algorithm for the monotone complementarity problem. *Math. Programming*, 84(2):375–399, February 1999.

[And09] Erling D. Andersen. The homogeneous and self-dual model and algorithm for linear optimization. Technical Report TR-1-2009, MOSEK ApS, 2009. URL: http://docs.mosek.com/whitepapers/homolo.pdf.

[And13] Erling D. Andersen. On formulating quadratic functions in optimization models. Technical Report TR-1-2013, MOSEK ApS, 2013. Last revised 23-feb-2016. URL: http://docs.mosek.com/whitepapers/qmodel.pdf.

[Naz87] J. L. Nazareth. *Computer Solution of Linear Programs*. Oxford University Press, New York, 1987.

[Wol98] L. A. Wolsey. *Integer programming*. John Wiley and Sons, 1998.

[MOSEKApS12] MOSEK ApS. *The MOSEK Modeling Cookbook*. MOSEK ApS, Fruebjergvej 3, Boks 16, 2100 Copenhagen O, 2012. Last revised September 2015. URL: http://docs.mosek.com/generic/modeling-a4.pdf.

Enumerations

Functions

Parameters

Response codes

Structures

Types