MOSEK

MOSEK ApS, C/O Symbion Science Park, Fruebjergvej 3, Box 16, 2100 Copenhagen Ø, Denmark

Contents

1 Introduction

MOSEK is a software package for the solution of linear, mixed-integer linear, quadratic, mixed-integer quadratic, quadratically constraint, and convex nonlinear mathematical optimization problems. MOSEK is particularly well suited for solving large-scale linear and convex quadratically constraint programs using an extremely efficient interior point algorithm. The interior point algorithm has many complex solver options which the user can specify to fine-tune the optimizer for a particular model.

Furthermore, MOSEK can solve generalized linear programs involving nonlinear conic constraints, convex quadratically constraint and general convex nonlinear programs.

These problem classes can be solved using an appropriate optimizer built into MOSEK. All the optimizers available in MOSEK are built for the solution of large-scale sparse problems. Current optimizers include:

- Interior-point optimizer for all continuous problems
- Conic interior-point optimizer for conic quadratic problems
- Simplex optimizer for linear problems
- Mixed-integer optimizer based on a branch and cut technology

1.1 Licensing

Licensing of GAMS/MOSEK is similar to other GAMS solvers. MOSEK is licensed in three different ways:

- **GAMS/MOSEK Base:**
  All continuous models
- **GAMS/MOSEK Extended:**
  Same as GAMS/MOSEK Base, but also the solution of models involving discrete variables.
- **GAMS/MOSEK Solver Link:**
  Users must have a seperate, licensed MOSEK system. For users who wish to use MOSEK within GAMS and also in other environments.

For more information contact `sales@gams.com`. For information regarding MOSEK standalone or interfacing MOSEK with other applications contact `sales@mosek.com`.

1.2 Reporting of Infeasible/Undbounded Models

MOSEK determines if either the primal or the dual problem is infeasible by means of a Farkas certificate. In such a case MOSEK returns a certificate indicating primal or dual infeasibility. A primal infeasibility certificate indicates a primal infeasible model and the certificate is reported in the marginals of the equations in the listing file. The primal infeasibility certificate for a minimization problem

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

is the solution y satisfying:

$$A^T y \leq 0, \quad b^T y > 0$$

A dual infeasibility certificate is reported in the levels of the variables in the listing file. The dual infeasibility certificate x for the same minimization problem is

$$Ax = 0, \quad c^T x < 0$$

Since GAMS reports all model statuses in the primal space, the notion of dual infeasibility does not exist and GAMS reports a status of unbounded, which assumes the primal problem is feasible. Although GAMS reports the primal as unbounded, there is the possibility that both the primal *and* dual problem are infeasible. To check if this is the case, the user can set appropriate upper and lower bounds on the objective variable, using the `(varible).LO` and `(variable).UP` suffixes and resolve.

For more information on primal and dual infeasibility certificates see the MOSEK User's manual at `www.mosek.com`.

1.3 Solving Problems in Parallel

If a computer has multiple CPUs (or a CPU with multiple cores), then it might be advantageous to use the multiple CPUs to solve the optimization problem. For instance if you have two CPUs you may want to exploit the two CPUs to solve the problem in the half time. MOSEK can exploit multiple CPUs.

Parallelized Optimizers

Only the interior-point optimizer in MOSEK has been parallelized.

This implies that whenever the MOSEK interior-point optimizer should solve an optimization problem, then it will try to divide the work so each CPU gets a share of the work. The user decides how many CPUs MOSEK should exploit. Unfortunately, it is not always easy to divide the work. Also some of the coordination work must occur in sequential. Therefore, the speed-up obtained when using multiple CPUs is highly problem dependent. However, as a rule of thumb if the problem solves very quickly i.e. in less than 60 seconds, then it is not advantageous of using the parallel option.

The parameter `MSK_IPAR_INTPNT_NUM_THREADS` sets the number of threads (and therefore the number of CPU's) that the interior point optimizer will use.

Concurrent Optimizer

An alternative to use a parallelized optimizer is the concurrent optimizer. The idea of the concurrent optimizer is to run multiple optimizers on the same problem concurrently. For instance the interior-point and the dual simplex optimizers may be applied to an linear optimization problem concurrently. The concurrent optimizer terminates when the first optimizer has completed and reports the solution of the fastest optimizer. That way a new optimizer has been created which essentially has the best performance of the interior-point and the dual simplex optimizer.

Hence, the concurrent optimizer is the best one to use if there multiple optimizers available in MOSEK for the problem and you cannot say beforehand which one is the best one. For more details inspect the `MSK_IPAR_CONCURRENT_*` options.

1.4 The Infeasibility Report

MOSEK has some facilities for diagnosing the cause of a primal or dual infeasibility. They can be turned on using the parameter setting MSK_IPAR_INFEAS_REPORT_AUTO. This causes MOSEK to print a report about an infeasible subset of the constraints, when an infeasibility is encountered. Moreover, the parameter MSK_IPAR_INFEAS_REPORT_LEVEL controls the amount info presented in the infeasibility report. We will use the trnsport.gms example from the GAMS Model Library with increased demand (b(j)=1.6*b(j)) to make the model infeasible. MOSEK produces the following infeasibility report

```
MOSEK PRIMAL INFEASIBILITY REPORT.

Problem status: The problem is primal infeasible

The following constraints are involved in the primal infeasibility.

Index     Name                    Lower bound      Upper bound      Dual lower       Dual upper
1         supply(seattle)         none             3.500000e+002    0.000000e+000    1.000000e+000
2         supply(san-diego)       none             6.000000e+002    0.000000e+000    1.000000e+000
3         demand(new-york)        5.200000e+002    none             1.000000e+000    0.000000e+000
4         demand(chicago)         4.800000e+002    none             1.000000e+000    0.000000e+000
5         demand(topeka)          4.400000e+002    none             1.000000e+000    0.000000e+000

The following bound constraints are involved in the infeasibility.

Index     Name                    Lower bound      Upper bound      Dual lower       Dual upper
```

which indicates which constraints and bounds that are important for the infeasibility i.e. causing the infeasibility. The infeasibility report is divided into two sections where the first section shows which constraints that are important for the infeasibility. In this case the important constraints are `supply` and `demand`. The values in the columns `Dual lower` and `Dual upper` are also useful, because if the dual lower value is different from zero for a constraint, then it implies that the lower bound on the constraint is important for the infeasibility. Similarly, if the dual upper value is different from zero on a constraint, then this implies the upper bound on the constraint is important for infeasibility.

1.5 Nonlinear Programs

MOSEK can efficiently solve convex programs, but is not intended for nonconvex optimization. For nonconvex programs, MOSEK can detect some nonconvexities and will print out a warning message and terminate. If MOSEK does not detect nonconvexities for a nonconvex model, the optimizer may continue but stagnate. Hence care must be taken when solving nonlinear programs if convexity is not immediately known.

1.6 Modeling Issues Involving Convex Programs

It is often preferable to model convex programs in seperable form, if it is possible. Consider the following example of minizing an objective function $f(x)$:

$$f(x) = \log(a' * x)$$

where $a \in \Re^n$ is a parameter and $x \in \Re^n$ the decision variable. The equation implies an implicit constraint of $a' * x > 0$. Unfortunately, domain violations can still occur because no restrictions are set on $a' * x$. A better approach is to introduce an intermediate variable y:

$$\begin{array}{ccc} f(x) & = & \log(y) \\ y & = & a' * x \\ y & \geq & 0 \end{array}$$

This accomplishes two things. It implies an explicit bound on $a' * x$, thereby reducing the risk of domain violations. Secondly, it speeds up computation since computations of gradients and Hessians in the first (non-seperable) form are more expensive. Finally, it reduces the amount of memory needed (see the section on “Memory Options”)

2 Conic Programming

MOSEK is well suited for solving generalized linear programs involving nonlinear conic constraints. Conic programming is useful in a wide variety of application areas[1] including engineering and financial management . Conic programming has been used, for example, in antenna array weight design, grasping force optimization, finite impulse response (FIR) filter design, and portfolio optimization.

This section gives an overview of conic programming and how conic constraints are implemented in GAMS.

2.1 Introduction

Conic programs can be thought of as generalized linear programs with the additional nonlinear constraint $x \in C$, where C is required to be a convex cone. The resulting class of problems is known as *conic optimization* and has the following form:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \leq r^c, \\ & x \in [l^x, u^x] \\ & x \in C \end{array}$$

where $A \in \Re^{m \times n}$ is the constraint matrix, $x \in \Re^n$ the decision variable, and $c \in \Re^n$ the objective function cost coefficients. The vector $r^c \in \Re^m$ represents the right hand side and the vectors $l^x, u^x \in \Re^n$ are lower and upper bounds on the decision variable x.

[1]See M. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, *Applications of second-order cone programming* Linear Algebra and its Applications, 284:193-228, Special Issue on Linear Algebra in Control, Signals and Image Processing. November, 1998.

Now partition the set of decision variables x into sets $S^t, t = 1, ..., k$, such that each decision variables x is a member of at most one set S^t. For example, we could have

$$S^1 = \begin{bmatrix} x_1 \\ x_4 \\ x_7 \end{bmatrix} \; and \; S^2 = \begin{bmatrix} x_6 \\ x_5 \\ x_3 \\ x_2 \end{bmatrix}. \tag{1.1}$$

Let x_{S^t} denote the variables x belonging to set S^t. Then define

$$C := \{x \in \Re^n : x_{S^t} \in C_t, t = 1, ..., k\} \tag{1.2}$$

where C_t must have one of the following forms:

- Quadratic cone: (also referred to as Lorentz or ice cream cone)

$$C_t = \left\{ x \in \Re^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\}. \tag{1.3}$$

- Rotated quadratic cone: (also referred to as hyberbolic constraints)

$$C_t = \left\{ x \in \Re^{n^t} : 2x_1x_2 \geq \sum_{j=3}^{n^t} x_j^2, \; x_1, x_2 \geq 0 \right\}. \tag{1.4}$$

These two types of cones allow the formulation of quadratic, quadratically constrained, and many other classes of nonlinear convex optimization problems.

2.2 Implemention of Conic Constraints in GAMS

GAMS handles conic equations using the `=C=` equation type. The conic cases are written as:

- Quadratic cone:

$$\texttt{x('1') = C = sum(i\$[not sameas(i,'1')], x(i));} \tag{1.5}$$

- Rotated quadratic cone:

$$\texttt{x('1') + x('2') = C = sum(i\$[not sameas(i,'1') and not sameas(i,'2')], x(i));} \tag{1.6}$$

Note that the resulting nonlinear conic constraints result in “linear” constraints in GAMS. Thus the original nonlinear formulation is in fact a linear model in GAMS. We remark that we could formulate conic problems as regular NLP using constraints:

- Quadratic cone:

$$\texttt{x('1') = G = sqrt[sum(i\$[not sameas(i,'1')], sqr[x(i)])];} \tag{1.7}$$

- Rotated quadratic cone: $x('1')$ and $x('2')$ are positive variables

$$\texttt{2 * x('1') * x('2') = G = sum(i\$[not sameas(i,'1') and not sameas(i,'2')], sqr[x(i)]);} \tag{1.8}$$

The example below illustrates the different formulations for conic programming problems. Note that the conic optimizer in MOSEK usually outperforms a general NLP method for the reformulated (NLP) cone problems.

2.3 Example

Consider the following example (`cone2.gms`) which illustrates the use of rotated conic constraints. We will give reformulations of the original problem in regular NLP form using conic constraints and in conic form.

The original problem is:

$$\begin{array}{lllr} \text{minimize} & \sum_i \frac{d_i}{x_i} & & \\ \text{subject to} & a^t x & \leq & b \\ & x_i & \in & [l_i, u_i], \quad l_i > 0, \;\; d_i \geq 0, \;\; i = 1, 2, ..., n \end{array} \tag{1.9}$$

where $x \in \Re^n$ is the decision variable, $d, a, l, u \in \Re^n$ parameters, and $b \in \Re$ a scalar parameter. The original model (1.9) can be written in GAMS using the equations:

```
defobj..      sum(n, d(n)/x(n)) =E= obj;
e1..          sum(n, a(n)*x(n)) =L= b;
Model orig /defobj, e1/;
x.lo(n) = l(n);
x.up(n) = u(n);
```

We can write an equivalent NLP formulation, replacing the objective function and adding another constraint:

$$\begin{array}{llll} \text{minimize} & \sum_i d_i t_i & & \\ \text{subject to} & a^t x & \leq & b \\ & 2 t_i x_i & \geq & 2, \qquad i = 1, ..., n \\ & x & \in & [l, u], \quad l > 0, \;\; d_i \geq 0 \end{array} \tag{1.10}$$

where $t \in \Re^n$ is a new decision variable. The GAMS formulation of this NLP (`model cnlp`) is:

```
defobjc..     sum(n, d(n)*t(n)) =E= obj;
e1..          sum(n, a(n)*x(n)) =L= b;
conenlp(n)..  2*t(n)*x(n) =G= 2;

Model cnlp /defobjc, e1, conenlp/;
x.lo(n) = l(n);
x.up(n) = u(n);
```

We can change the equality to an inequality since the parameter $d_i \geq 0$ and we are dealing with a minimization problem. Also, note that the constraint `conenlp(n)` is almost in rotated conic form. If we introduce a variable $z \in \Re^n, z_i = \sqrt{2}$, then we can reformulate the problem using conic constraints as:

$$\begin{array}{llll} \text{minimize} & \sum_i d_i t_i & & \\ \text{subject to} & a^t x & \leq & b \\ & z_i & = & \sqrt{2} \\ & 2 t_i x_i & \geq & z_i^2, \qquad i = 1, ..., n \\ & x & \in & [l, u], \quad l > 0, \;\; d_i \geq 0 \end{array} \tag{1.11}$$

The GAMS formulation using conic equations `=C=` is:

```
defobjc..     sum(n, d(n)*t(n)) =E= obj;
e1..          sum(n, a(n)*x(n)) =L= b;
e2(n)..       z(n) =E= sqrt(2);
cone(n)..     x(n) + t(n) =C= z(n);

Model clp  /defobjc, e1, e2, cone/;
x.lo(n) = l(n);
x.up(n) = u(n);
```

Note that this formulation is a linear program in GAMS, although the constraints `cone(n)...` represent the nonlinear rotated quadratic cone constraint.

The complete model is listed below:

```
Set n / n1*n10 /;
Parameter d(n), a(n), l(n), u(n);
Scalar b;

d(n) = uniform(1,2);
a(n) = uniform (10,50);
l(n) = uniform(0.1,10);
u(n) = l(n) + uniform(0,12-l(n));

Variables x(n);
x.l(n) = uniform(l(n), u(n));
b = sum(n, x.l(n)*a(n));

Variables t(n), z(n), obj;
Equations defobjc, defobj, e1, e2(n), cone(n), conenlp(n);

defobjc..     sum(n, d(n)*t(n)) =E= obj;
defobj..      sum(n, d(n)/x(n)) =E= obj;
e1..          sum(n, a(n)*x(n)) =L= b;
e2(n)..       z(n) =E= sqrt(2);
cone(n)..     x(n) + t(n) =C= z(n);
conenlp(n)..  2*t(n)*x(n) =G= 2;

Model clp  /defobjc, e1, e2, cone/;
Model cnlp /defobjc, e1, conenlp/;
Model orig /defobj, e1/;

x.lo(n) = l(n);
x.up(n) = u(n);

Solve clp  min obj using lp;
Solve cnlp min obj using nlp;
Solve orig min obj using nlp;
```

3 The MOSEK Options

MOSEK works like other GAMS solvers, and many options can be set in the GAMS model. The most relevant GAMS options are `reslim, nodlim, optca, optcr`, and `optfile`. The option `iterlim` works only for the simplex optimizer. A description of all available GAMS options can be found in Chapter "Using Solver Specific Options".

We remark that MOSEK contains many complex solver options, many of which require a deep understanding of the algorithms used. For a complete description of the more than 175 MOSEK options, consult the MOSEK User's Guide, available online at `www.mosek.com`.

If you specify "`<modelname>.optfile = 1;`" before the SOLVE statement in your GAMS model, MOSEK will then look for and read an option file with the name *mosek.opt* (see "Using Solver Specific Options" for general use of solver option files). The syntax for the MOSEK option file is

```
optname value
```

with one option on each line.

For example,

```
MSK_IPAR_INTPNT_MAX_ITERATIONS 20
MSK_IPAR_INTPNT_SCALING         1
```

The first option specifies the maximum number of interior-point iterations, in this case 20. The seond option indicates a scaling option of 1, which is no scaling.

We remark that users can also use symbolic constants in place of numerical values. For example, for the scaling option users could use `MSK_SCALING_NONE` in place of the value 1. For a complete list of applicable symbolic constants, consult the MOSEK parameter list available online at `www.mosek.com`.

3.1 Memory Considerations for Nonlinear Problems

The GAMS *workfactor* option can be used to increase the amount of memory available to MOSEK. The general syntax is

(modelname).`workfactor` = *(value)*

with a default value of 1. See the section on "Using Solver Specific Options" for details. If GAMS/MOSEK runs out of memory, an error message is printed out:

```
*** GAMS/MOSEK interface error.

        The size estimate for Hessian of Lagrangian is too small.
        Try to increase workfactor option from 1 to a larger value.
```

GAMS/MOSEK estimates the size of the Hessian as $5*(number\ of\ nonlinear\ variables)*(workfactor)$. Because of symmetry, the size of the Hessian is bounded by

$$H_d * (number\ of\ nonlinear\ variables)^2/2$$

where H_d detotes the density of the Hessian and $H_d \in [0,1]$. Therefore, one can choose the workfactor as:

$$workfactor = H_d * (number\ of\ nonlinear\ variables) * /(5*2)$$

Note that for a seperable model (see "Modeling Issues Involving Convex Programs"), the workfactor can in fact be reduced to 1/5.

4 Summary of MOSEK Options

4.1 General and Preprocessing Options

MSK_IPAR_OPTIMIZER	optimizer selection
MSK_DPAR_OPTIMIZER_MAX_TIME	time limit
MSK_IPAR_PRESOLVE_ELIMINATOR_USE	switch for free variable elimination
MSK_IPAR_PRESOLVE_ELIM_FILL	fill-in control during presolve
MSK_IPAR_PRESOLVE_LINDEP_USE	linear dependency check
MSK_IPAR_PRESOLVE_LINDEP_WORK_LIM	maximum work for finding linear dependencies
MSK_IPAR_PRESOLVE_USE	switch for presolve
MSK_IPAR_CACHE_SIZE_L1	L1 cache size used
MSK_IPAR_CACHE_SIZE_L2	

L2 cache size used

MSK_IPAR_CPU_TYPE
specifies the CPU type

MSK_SPAR_PARAM_READ_FILE_NAME
name of a secondary MOSEK option file

MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS
maximum number of optimizers during concurrent run

MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX
priority of dual simplex algorithm in concurrent run

MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX
priority of free simplex algorithm in concurrent run

MSK_IPAR_CONCURRENT_PRIORITY_INTPNT
priority of interior point algorithm in concurrent run

MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX
priority of primal simplex algorithm in concurrent run

MSK_IPAR_INFEAS_REPORT_AUTO
switch for infeasibility report

MSK_IPAR_INFEAS_REPORT_LEVEL
output level for infeasibility report

4.2 Problem Data Options

MSK_DPAR_DATA_TOL_AIJ
zero tolerance for matrix coefficients

MSK_DPAR_DATA_TOL_AIJ_LARGE
warning for large coefficients in matrix

MSK_DPAR_DATA_TOL_BOUND_INF
bound value for infinity

MSK_DPAR_DATA_TOL_BOUND_WRN
warning for large bounds

MSK_DPAR_DATA_TOL_CJ_LARGE
warning for large coefficients in objective

MSK_DPAR_DATA_TOL_C_HUGE
error for huge coefficients in objective

MSK_DPAR_DATA_TOL_QIJ
zero tolerance for Q matrix coefficients

MSK_DPAR_DATA_TOL_X
tolerance for fixed variables

MSK_IPAR_CHECK_CONVEXITY
level of convexity check for quadratic problems

MSK_DPAR_LOWER_OBJ_CUT
lower objective limit

MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH
upper objective limit threashold

MSK_DPAR_UPPER_OBJ_CUT
upper objective limit

MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH
lower objective limit threashold

4.3 Output Options

MSK_IPAR_LOG_BI
: output control for basis identification

MSK_IPAR_LOG_INTPNT
: output level of the interior-point optimizer

MSK_IPAR_LOG_MIO
: output level for mixed integer optimizer

MSK_IPAR_LOG_PRESOLVE
: output level for presolve

MSK_IPAR_LOG_SIM
: output level for simplex

MSK_IPAR_LOG_BI_FREQ
: frequency of log output of basis identification

MSK_IPAR_LOG_SIM_FREQ
: frequency of log output of simplex optimizer

MSK_IPAR_LOG_MIO_FREQ
: frequency of log output of mixed integer optimizer

MSK_IPAR_WARNING_LEVEL
: warning level

MSK_IPAR_MAX_NUM_WARNINGS
: maximum number of warnings

4.4 Interior Point Optimizer Options

MSK_IPAR_INTPNT_BASIS
: switch for basis identification

MSK_DPAR_INTPNT_CO_TOL_DFEAS
: dual feasibility tolerance for the conic interior-point optimizer

MSK_DPAR_INTPNT_CO_TOL_INFEAS
: infeasibility control for the conic interior-point optimizer

MSK_DPAR_INTPNT_CO_TOL_MU_RED
: relative complementarity tolerance for the conic interior-point optimizer

MSK_DPAR_INTPNT_CO_TOL_NEAR_REL
: termination tolerances for near optimal for the conic interior-point optimizer

MSK_DPAR_INTPNT_CO_TOL_PFEAS
: primal feasibility tolerance for the conic interior-point optimizer

MSK_DPAR_INTPNT_CO_TOL_REL_GAP
: relative optimality tolerance for the conic interior-point optimizer

MSK_IPAR_INTPNT_DIFF_STEP
: switch for different step sizes

MSK_IPAR_INTPNT_MAX_ITERATIONS
: iteration limit for the interior-point optimizer

MSK_DPAR_INTPNT_NL_MERIT_BAL
: balance for complementarity and infeasibility

MSK_DPAR_INTPNT_NL_TOL_DFEAS
: dual feasibility tolerance for nonlinear problems

MSK_DPAR_INTPNT_NL_TOL_MU_RED
: relative complementarity tolerance for nonlinear problems

MSK_DPAR_INTPNT_NL_TOL_NEAR_REL
: termination tolerances for near optimal for nonlinear problems

MSK_DPAR_INTPNT_NL_TOL_PFEAS
: primal feasibility tolerance for nonlinear problems

MSK_DPAR_INTPNT_NL_TOL_REL_GAP
: relative optimality tolerance for nonlinear problems

MSK_DPAR_INTPNT_NL_TOL_REL_STEP	relative step size to boundary for nonlinear problems
MSK_IPAR_INTPNT_NUM_THREADS	number of threads for interior-point optimizer
MSK_IPAR_INTPNT_OFF_COL_TRH	offending column selection
MSK_IPAR_INTPNT_ORDER_METHOD	ordering strategy selection
MSK_IPAR_INTPNT_REGULARIZATION_USE	switch for regularization
MSK_IPAR_INTPNT_SCALING	scaling selection for interior-point optimizer
MSK_IPAR_INTPNT_SOLVE_FORM	solve primal or the dual problem with interior-point optimizer
MSK_IPAR_INTPNT_STARTING_POINT	starting point for interior-point optimizer
MSK_DPAR_INTPNT_TOL_DFEAS	dual feasibility tolerance
MSK_DPAR_INTPNT_TOL_DSAFE	initial dual starting control
MSK_DPAR_INTPNT_TOL_INFEAS	infeasibility control
MSK_DPAR_INTPNT_TOL_MU_RED	relative complementarity tolerance
MSK_DPAR_INTPNT_TOL_PATH	central path following for interior-point optimizer
MSK_DPAR_INTPNT_TOL_PFEAS	primal feasibility tolerance
MSK_DPAR_INTPNT_TOL_PSAFE	initial primal starting control
MSK_DPAR_INTPNT_TOL_REL_GAP	relative optimality tolerance
MSK_DPAR_INTPNT_TOL_REL_STEP	relative step size to boundary
MSK_IPAR_INTPNT_MAX_NUM_COR	maximum number of correctors
MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS	number of steps to be used by the iterative refinement
USE_BASIS_EST	use MOSEK basis estimation in case of an interior solution

4.5 Simplex Optimizer and Basis Identification Options

MSK_IPAR_SIM_DUAL_CRASH	dual simplex crash
MSK_IPAR_SIM_DUAL_SELECTION	dual simplex pricing selection
MSK_IPAR_SIM_HOTSTART	controls simplex hotstart
MSK_IPAR_SIM_MAX_ITERATIONS	simplex iteration limit
MSK_IPAR_SIM_MAX_NUM_SETBACKS	maximum number of setbacks

MSK_IPAR_SIM_PRIMAL_CRASH
: primal simplex crash

MSK_IPAR_SIM_PRIMAL_SELECTION
: primal simplex pricing selection

MSK_IPAR_SIM_REFACTOR_FREQ
: refactorization frequency

MSK_IPAR_SIM_SCALING
: scaling selection for simplex optimizer

MSK_IPAR_SIM_SOLVE_FORM
: solve primal or the dual problem with simplex optimizer

MSK_IPAR_BI_CLEAN_OPTIMIZER
: simplex optimizer section after basis identification

MSK_DPAR_BI_LU_TOL_REL_PIV
: relative pivot tolerance for basis identification

MSK_IPAR_BI_MAX_ITERATIONS
: maximum number of simplex iterations after basis identification

MSK_IPAR_BI_IGNORE_MAX_ITER
: continues BI in case of iteration limit

MSK_IPAR_BI_IGNORE_NUM_ERROR
: continues BI in case of numerical error

4.6 Mixed Integer Optimizer Options

MSK_IPAR_MIO_BRANCH_DIR
: control branching directions

MSK_IPAR_MIO_CONSTRUCT_SOL
: switch for mip start

MSK_IPAR_MIO_CUT_LEVEL_ROOT
: cut level control at root for mixed integer optimizer

MSK_IPAR_MIO_HEURISTIC_LEVEL
: heuristic control for mixed integer optimizer

MSK_DPAR_MIO_HEURISTIC_TIME
: time limit for heuristic search

MSK_IPAR_MIO_KEEP_BASIS
: switch for basis saving

MSK_IPAR_MIO_MAX_NUM_BRANCHES
: maximum number of branches

MSK_IPAR_MIO_MAX_NUM_RELAXS
: maximum number of relaxations solved

MSK_DPAR_MIO_MAX_TIME
: time limit for mixed integer optimizer

MSK_DPAR_MIO_MAX_TIME_APRX_OPT
: time limit before some relaxation

MSK_DPAR_MIO_NEAR_TOL_ABS_GAP
: termination criterion on absolute optimality tolerance

MSK_DPAR_MIO_NEAR_TOL_REL_GAP
: termination criterion on relative optimality tolerance

MSK_IPAR_MIO_NODE_OPTIMIZER
: solver for the sub problems

MSK_IPAR_MIO_NODE_SELECTION
: node selection strategy

MSK_IPAR_MIO_PRESOLVE_AGGREGATE
: switch for aggregation during mixed integer presolve

MSK_IPAR_MIO_PRESOLVE_USE	switch for mixed integer presolve
MSK_DPAR_MIO_REL_ADD_CUT_LIMITED	cuts factor
MSK_IPAR_MIO_ROOT_OPTIMIZER	solver for the root problem
MSK_IPAR_MIO_STRONG_BRANCH	strong branching control
MSK_DPAR_MIO_TOL_ABS_GAP	absolute optimality tolerance in the mixed integer optimizer
MSK_DPAR_MIO_TOL_ABS_RELAX_INT	absolute integrality tolerance
MSK_DPAR_MIO_TOL_REL_GAP	relative optimality tolerance in the mixed integer optimizer
MSK_DPAR_MIO_TOL_REL_RELAX_INT	relative integrality tolerance
MSK_IPAR_MIO_PRESOLVE_PROBING	switch for probing
MSK_IPAR_MIO_CUT_LEVEL_TREE	cut level control in tree for mixed integer optimizer

5 Detailed Descriptions of MOSEK Options

MSK_IPAR_OPTIMIZER *(integer)*

Controls which optimizer is used to optimize the task.

(default = 0)

0 The choice of optimizer is made automatically.
1 The interior-point optimizer is used.
2 Another cone optimizer.
3 The Qcone optimizer is used.
4 The primal simplex optimizer is used.
5 The dual simplex optimizer is used.
6 Either the primal or the dual simplex optimizer is used.
7 The mixed integer optimizer.
8 The optimizer for nonconvex nonlinear problems.
9 The concurrent optimizer is used.

MSK_DPAR_OPTIMIZER_MAX_TIME *(real)*

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

(default = GAMS ResLim)

MSK_IPAR_PRESOLVE_ELIMINATOR_USE *(integer)*

Controls whether free or implied free variables are eliminated from the problem.

(default = 1)

MSK_IPAR_PRESOLVE_ELIM_FILL *(integer)*

Controls the maximum amount of fill-in that can be created during the eliminations phase of the presolve. This parameter times the number of variables plus the number of constraints denotes the amount of fill in.

(default = 1)

MSK_IPAR_PRESOLVE_LINDEP_USE (*integer*)

Controls whether the linear constraints is checked for linear dependencies.

(default = 1)

MSK_IPAR_PRESOLVE_LINDEP_WORK_LIM (*integer*)

Is used to limit the work that can used to locate the linear dependencies. In general the higher value this parameter is given the less work can be used. However, a value of 0 means no limit on the amount work that can be used.

(default = 1)

MSK_IPAR_PRESOLVE_USE (*integer*)

Controls whether presolve is performed.

(default = 2)

0 The problem is not presolved before it is optimized

1 The problem is presolved before it is optimized.

2 It is decided automatically whether to presolve before the problem is optimized.

MSK_IPAR_CACHE_SIZE_L1 (*integer*)

Controls the size of the L1 cache used by MOSEK.

(default = -1)

MSK_IPAR_CACHE_SIZE_L2 (*integer*)

Controls the size of the L2 cache used by MOSEK.

(default = -1)

MSK_IPAR_CPU_TYPE (*integer*)

This option specifies the CPU type.

(default = 0)

0 An unknown CPU.

1 An generic CPU type for the platform.

2 An Intel Pentium P3.

3 An Intel Pentium P4 or Intel Xeon.

4 An AMD Athlon.

5 A HP PA RISC version 2.0 CPU.

6 An Intel Itanium2.

7 An AMD Opteron (64 bit).

8 A G5 PowerPC CPU.

9 An Intel PM CPU.

10 An Intel CORE2 cpu.

MSK_SPAR_PARAM_READ_FILE_NAME (*string*)

The name of a secondary MOSEK option file that by the MOSEK option reader.

MSK_IPAR_CONCURRENT_NUM_OPTIMIZERS (*integer*)

The maximum number of simultaneous optimizations that will be started by the concurrent optimizer.

(default = 2)

MSK_IPAR_CONCURRENT_PRIORITY_DUAL_SIMPLEX (*integer*)

Priority of the dual simplex algorithm when selecting solvers for concurrent optimization.

(default = 2)

MSK_IPAR_CONCURRENT_PRIORITY_FREE_SIMPLEX (*integer*)

Priority of free simplex optimizer when selecting solvers for concurrent optimization.

(default = 3)

MSK_IPAR_CONCURRENT_PRIORITY_INTPNT (*integer*)

Priority of the interior point algorithm when selecting solvers for concurrent optimization.

(default = 4)

MSK_IPAR_CONCURRENT_PRIORITY_PRIMAL_SIMPLEX (*integer*)

Priority of the primal simplex algorithm when selecting solvers for concurrent optimization.

(default = 1)

MSK_IPAR_INFEAS_REPORT_AUTO (*integer*)

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

(default = 0)

MSK_IPAR_INFEAS_REPORT_LEVEL (*integer*)

Controls the amount info presented in an infeasibility report. Higher values implies more information.

(default = 1)

MSK_DPAR_DATA_TOL_AIJ (*real*)

Absolute zero tolerance for coefficients in the constraint matrix.

Range: [1.0e-16,1.0e-6]

(default = 1.0e-12)

MSK_DPAR_DATA_TOL_AIJ_LARGE (*real*)

A coefficient in the constraint matrix which is larger than this value in absolute size causes a warning message to be printed.

(default = 1.0e10)

MSK_DPAR_DATA_TOL_BOUND_INF (*real*)

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

(default = 1.0e16)

MSK_DPAR_DATA_TOL_BOUND_WRN (*real*)

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

(default = 1.0e8)

MSK_DPAR_DATA_TOL_CJ_LARGE (*real*)

A coefficient in the objective which is larger than this value in absolute terms causes a warning message to be printed.

(default = 1.0e8)

MSK_DPAR_DATA_TOL_C_HUGE (*real*)

A coefficient in the objective which is larger than the value of this parameter in absolute terms is considered to be huge and generates an error.

(default = 1.0e16)

MSK_DPAR_DATA_TOL_QIJ (*real*)

Absolute zero tolerance for coefficients in the Q matrices.

(default = 1.0e-16)

MSK_DPAR_DATA_TOL_X (*real*)

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

(default = 1.0e-8)

MSK_IPAR_CHECK_CONVEXITY (*integer*)

Specify the level of convexity check on quadratic problems.

(default = 1)

0 No convexity check.

1 Perform simple and fast convexity check.

MSK_DPAR_LOWER_OBJ_CUT (*real*)

If a feasible solution having and objective value outside, the interval `LOWER_OBJ_CUT`, `UPPER_OBJ_CUT`, then MOSEK is terminated.

(default = -1.0e30)

MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH (*real*)

If the lower objective cut (`LOWER_OBJ_CUT`) is less than `LOWER_OBJ_CUT_FINITE_TRH`, then the lower objective cut `LOWER_OBJ_CUT` is treated as infinity.

(default = -0.5e30)

MSK_DPAR_UPPER_OBJ_CUT (*real*)

If a feasible solution having and objective value outside, the interval `LOWER_OBJ_CUT`, `UPPER_OBJ_CUT`, then MOSEK is terminated.

(default = 1.0e30)

MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH (*real*)

If the upper objective cut (`UPPER_OBJ_CUT`) is greater than `UPPER_OBJ_CUT_FINITE_TRH`, then the upper objective cut `UPPER_OBJ_CUT` is treated as infinity.

(default = 0.5e30)

MSK_IPAR_LOG_BI (*integer*)

Controls the amount of output printed by the basis identification procedure.

(default = 4)

MSK_IPAR_LOG_INTPNT (*integer*)

Controls the amount of output printed by the interior-point optimizer.

(default = 4)

MSK_IPAR_LOG_MIO (*integer*)

Controls the print level for the mixed integer optimizer.

(default = 4)

MSK_IPAR_LOG_PRESOLVE (*integer*)

Controls amount of output printed by the presolve procedure.

(default = 1)

MSK_IPAR_LOG_SIM (*integer*)

Controls amount of output printed by the simplex optimizer.

(default = 4)

MSK_IPAR_LOG_BI_FREQ (*integer*)

Controls how frequent the optimizer outputs information about the basis identification is called.

(default = 2500)

MSK_IPAR_LOG_SIM_FREQ (*integer*)

Controls how frequent the simplex optimizer outputs information about the optimization.

(default = 500)

MSK_IPAR_LOG_MIO_FREQ (*integer*)

Controls how frequent the mixed integer optimizer prints the log line. It will print a line every time `MSK_INTPAR_LOG_MIO_FREQ` relaxations have been solved.

(default = 250)

MSK_IPAR_WARNING_LEVEL (*integer*)

Warning level. A higher value implies more warnings.

(default = 1)

MSK_IPAR_MAX_NUM_WARNINGS (*integer*)

Sets the maximum number of warnings.

(default = 10)

MSK_IPAR_INTPNT_BASIS (*integer*)

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

(default = 1)

0 Never do basis identification.

1 Basis identification is always performed even if the interior-point optimizer terminates abnormally.

2 Basis identification is performed if the interior-point optimizer terminates without an error.

3 Basis identification is not performed if the interior-point optimizer terminates with a problem status saying that the problem is primal or dual infeasible.

4 Try another BI method.

MSK_DPAR_INTPNT_CO_TOL_DFEAS (*real*)

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

Range: [0.0,1.0]

(default = 1.0e-8)

MSK_DPAR_INTPNT_CO_TOL_INFEAS (*real*)

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.

Range: [0.0,1.0]

(default = 1.0e-8)

MSK_DPAR_INTPNT_CO_TOL_MU_RED (*real*)

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

Range: [0.0,1.0]

(default = 1.0e-8)

MSK_DPAR_INTPNT_CO_TOL_NEAR_REL (*real*)

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

(default = 100)

MSK_DPAR_INTPNT_CO_TOL_PFEAS (*real*)

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

Range: [0.0,1.0]

(default = 1.0e-8)

MSK_DPAR_INTPNT_CO_TOL_REL_GAP (*real*)

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

Range: [0.0,1.0]

(default = 1.0e-8)

MSK_IPAR_INTPNT_DIFF_STEP (*integer*)

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

(default = 1)

MSK_IPAR_INTPNT_MAX_ITERATIONS (*integer*)

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

(default = 400)

MSK_DPAR_INTPNT_NL_MERIT_BAL (*real*)

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

Range: [0.0,0.99]

(default = 1.0e-4)

MSK_DPAR_INTPNT_NL_TOL_DFEAS (*real*)

Dual feasibility tolerance used when a nonlinear model is solved.

Range: [0.0,1.0]

(default = 1.0e-8)

MSK_DPAR_INTPNT_NL_TOL_MU_RED (*real*)

Relative complementarity gap tolerance used when a nonlinear model is solved..

Range: [0.0,1.0]

(default = 1.0e-12)

MSK_DPAR_INTPNT_NL_TOL_NEAR_REL (*real*)

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

(default = 1000.0)

MSK_DPAR_INTPNT_NL_TOL_PFEAS (*real*)

Primal feasibility tolerance used when a nonlinear model is solved.

Range: [0.0,1.0]

(default = 1.0e-8)

MSK_DPAR_INTPNT_NL_TOL_REL_GAP (*real*)

Relative gap termination tolerance for nonlinear problems.

(default = 1.0e-6)

MSK_DPAR_INTPNT_NL_TOL_REL_STEP (*real*)

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

Range: [1.0e-4,0.9999999]

(default = 0.995)

MSK_IPAR_INTPNT_NUM_THREADS (*integer*)

Controls the number of threads employed by the interior-point optimizer.

(default = 1)

MSK_IPAR_INTPNT_OFF_COL_TRH (*integer*)

Controls how many offending columns there are located in the Jacobian the constraint matrix. 0 means no offending columns will be detected. 1 means many offending columns will be detected. In general by increasing the number fewer offending columns will be detected.

(default = 40)

MSK_IPAR_INTPNT_ORDER_METHOD (*integer*)

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

(default = 0)

0 The ordering method is automatically chosen.

1 Approximate minimum local-fill-in ordering is used.

2 A variant of the approximate minimum local-fill-in ordering is used.

3 Graph partitioning based ordering.

4 An alternative graph partitioning based ordering.

5 No ordering is used.

MSK_IPAR_INTPNT_REGULARIZATION_USE (*integer*)

Controls whether regularization is allowed.

(default = 1)

MSK_IPAR_INTPNT_SCALING (*integer*)

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

(default = 0)

0 The optimizer choose the scaling heuristic.

1 No scaling is performed.

2 A conservative scaling is performed.

3 A very aggressive scaling is performed.

MSK_IPAR_INTPNT_SOLVE_FORM (*integer*)

Controls whether the primal or the dual problem is solved.

(default = 0)

0 The optimizer is free to solve either the primal or the dual problem.

1 The optimizer should solve the primal problem.

2 The optimizer should solve the dual problem.

MSK_IPAR_INTPNT_STARTING_POINT (*integer*)

Selection of starting point used by the interior-point optimizer.

(default = 0)

0 The starting point is chosen automatically.

1 The starting point is chosen to be constant. This is more reliable than a non-constant starting point.

MSK_DPAR_INTPNT_TOL_DFEAS (*real*)

Dual feasibility tolerance used for linear and quadratic optimization problems.

Range: [0.0,1.0]

(default = 1.0e-8)

MSK_DPAR_INTPNT_TOL_DSAFE (*real*)

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

(default = 1.0)

MSK_DPAR_INTPNT_TOL_INFEAS (*real*)

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.

Range: [0.0,1.0]

(default = 1.0e-8)

MSK_DPAR_INTPNT_TOL_MU_RED (*real*)

Relative complementarity gap tolerance

Range: [0.0,1.0]

(default = 1.0e-16)

MSK_DPAR_INTPNT_TOL_PATH (*real*)

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

Range: [0.0,0.9999]

(default = 1.0e-8)

MSK_DPAR_INTPNT_TOL_PFEAS (*real*)

Primal feasibility tolerance used for linear and quadratic optimization problems.

Range: [0.0,1.0]

(default = 1.0e-8)

MSK_DPAR_INTPNT_TOL_PSAFE (*real*)

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

(default = 1.0)

MSK_DPAR_INTPNT_TOL_REL_GAP (*real*)

Relative gap termination tolerance.

(default = 1.0e-8)

MSK_DPAR_INTPNT_TOL_REL_STEP (*real*)

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

Range: [1.0e-4,0.999999]

(default = 0.9999)

MSK_IPAR_INTPNT_MAX_NUM_COR (*integer*)

Controls the maximum number of correctors allowed by the multiple corrector procedure. A negative value means that Mosek is making the choice.

(default = -1)

MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS (*integer*)

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

(default = -1)

USE_BASIS_EST (*integer*)

(default = 0)

MSK_IPAR_SIM_DUAL_CRASH (*integer*)

Controls whether crashing is performed in the dual simplex optimizer. In general if a basis consists of more than `(100*SIM_DUAL_CRASH)` percent fixed variables, then a crash will be performed.

(default = GAMS BRatio)

MSK_IPAR_SIM_DUAL_SELECTION (*integer*)

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

(default = 0)

0	The optimizer choose the pricing strategy.
1	The optimizer uses full pricing.
2	The optimizer uses approximate steepest-edge pricing.
3	The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).
4	The optimizer uses an partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.
5	The optimizer uses a partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.

MSK_IPAR_SIM_HOTSTART (*integer*)

Controls whether the simplex optimizer will do hotstart if possible.

(default = 1)

0	The simplex optimizer performs a coldstart.
1	The simplex optimize chooses the hot-start type.
2	Only the status keys of the constraints and variables are used to choose the type of hot-start.

MSK_IPAR_SIM_MAX_ITERATIONS (*integer*)

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

(default = GAMS IterLim)

MSK_IPAR_SIM_MAX_NUM_SETBACKS (*integer*)

Controls how many setbacks that are allowed within a simplex optimizer. A setback is an event where the optimizer moves in the wrong direction. This is impossible in theory but may happen due to numerical problems.

(default = 250)

MSK_IPAR_SIM_PRIMAL_CRASH *(integer)*

Controls whether crashing is performed in the primal simplex optimizer. In general if a basis consists of more than `(100*SIM_PRIMAL_CRASH)` percent fixed variables, then a crash will be performed.

(default = GAMS BRatio)

MSK_IPAR_SIM_PRIMAL_SELECTION *(integer)*

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

(default = 0)

0 The optimizer choose the pricing strategy.

1 The optimizer uses full pricing.

2 The optimizer uses approximate steepest-edge pricing.

3 The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

4 The optimizer uses an partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.

5 The optimizer uses a partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.

MSK_IPAR_SIM_REFACTOR_FREQ *(integer)*

Controls how frequent the basis is refactorized. The value 0 means that the optimizer determines when the best point of refactorization is.

(default = 0)

MSK_IPAR_SIM_SCALING *(integer)*

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

(default = 0)

0 The optimizer choose the scaling heuristic.

1 No scaling is performed.

2 A conservative scaling is performed.

3 A very aggressive scaling is performed.

MSK_IPAR_SIM_SOLVE_FORM *(integer)*

Controls whether the primal or the dual problem is solved by the simplex optimizers.

(default = 0)

0 The optimizer is free to solve either the primal or the dual problem.

1 The optimizer should solve the primal problem.

2 The optimizer should solve the dual problem.

MSK_IPAR_BI_CLEAN_OPTIMIZER *(integer)*

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

(default = 0)

0 The choice of optimizer is made automatically.

1 The interior-point optimizer is used.

2 Another cone optimizer.

3 The Qcone optimizer is used.

4 The primal simplex optimizer is used.

5 The dual simplex optimizer is used.

6 Either the primal or the dual simplex optimizer is used.

7 The mixed integer optimizer.

8 The optimizer for nonconvex nonlinear problems.

9 The concurrent optimizer is used.

MSK_DPAR_BI_LU_TOL_REL_PIV (*real*)

Relative pivot tolerance used in the LU factorization in the basis identification procedure.

Range: [1.0e-6,0.999999]

(default = 0.01)

MSK_IPAR_BI_MAX_ITERATIONS (*integer*)

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

(default = 1000000)

MSK_IPAR_BI_IGNORE_MAX_ITER (*integer*)

If the parameter MSK_IPAR_INTPNT_BASIS has the value 2 and the interior-point optimizer has terminated due to maximum number of iterations, then basis identification is performed if this parameter has the value 1.

(default = 0)

MSK_IPAR_BI_IGNORE_NUM_ERROR (*integer*)

If the parameter MSK_IPAR_INTPNT_BASIS has the value 2 and the interior-point optimizer has terminated due to a numerical problem, then basis identification is performed if this parameter has the value 1.

(default = 0)

MSK_IPAR_MIO_BRANCH_DIR (*integer*)

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

(default = 0)

0 The mixed optimizer decides which branch to choose.

1 The mixed integer optimizer always chooses the up branch.

2 The mixed integer optimizer always chooses the down branch.

MSK_IPAR_MIO_CONSTRUCT_SOL (*integer*)

If set to 1 and all integer variables has been given a value for which a feasible MIP solution exists, then MOSEK generates an initial solution to the MIP by fixing all integer values and solving for the continues variables.

(default = 0)

MSK_IPAR_MIO_CUT_LEVEL_ROOT (*integer*)

Controls the cut level employed by the mixed integer optimizer. A negative value means a default value determined by the mixed integer optimizer is used. By adding the appropriate values from the following table the employed cut types can be controlled.

GUB cover	+2
Flow cover	+4
Lifting	+8
Plant location	+16
Disaggregation	+32
Knapsack cover	+64
Lattice	+128
Gomory	+256

```
Coefficient reduction  +512
GCD                    +1024
Obj. integrality       +2048
```

(default = -1)

MSK_IPAR_MIO_HEURISTIC_LEVEL (*integer*)

Controls the heuristic employed by the mixed integer optimizer to locate an integer feasible solution. A value of zero means no heuristic is used. A large value than 0 means a gradually more sophisticated heuristic is used which is computationally more expensive. A negative value implies that the optimizer chooses the heuristic to be used.

(default = -1)

MSK_DPAR_MIO_HEURISTIC_TIME (*real*)

Maximum time allowed to be used in the heuristic search for an optimal integer solution. A negative values implies that the optimizer decides the amount of time to be spend in the heuristic.

(default = -1.0)

MSK_IPAR_MIO_KEEP_BASIS (*integer*)

Controls whether the integer presolve keeps bases in memory. This speeds on the solution process at cost of bigger memory consumption.

(default = 1)

MSK_IPAR_MIO_MAX_NUM_BRANCHES (*integer*)

Maximum number branches allowed during the branch and bound search. A negative value means infinite.

(default = -1)

MSK_IPAR_MIO_MAX_NUM_RELAXS (*integer*)

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

(default = -1)

MSK_DPAR_MIO_MAX_TIME (*real*)

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

(default = -1.0)

MSK_DPAR_MIO_MAX_TIME_APRX_OPT (*real*)

Number of seconds spend by the mixed integer optimizer before the `MIO_TOL_REL_RELAX_INT` is applied.

(default = 60)

MSK_DPAR_MIO_NEAR_TOL_ABS_GAP (*real*)

Relaxed absolute optimality tolerance employed by the mixed integer optimizer. The mixed integer optimizer is terminated when this tolerance is satisfied.

(default = GAMS OptCa)

MSK_DPAR_MIO_NEAR_TOL_REL_GAP (*real*)

Relaxed relative optimality tolerance employed by the mixed integer optimizer. The mixed integer optimizer is terminated when this tolerance is satisfied.

(default = GAMS OptCr)

MSK_IPAR_MIO_NODE_OPTIMIZER (*integer*)

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

(default = 0)

0 The choice of optimizer is made automatically.
1 The interior-point optimizer is used.
2 Another cone optimizer.
3 The Qcone optimizer is used.
4 The primal simplex optimizer is used.
5 The dual simplex optimizer is used.
6 Either the primal or the dual simplex optimizer is used.
7 The mixed integer optimizer.
8 The optimizer for nonconvex nonlinear problems.
9 The concurrent optimizer is used.

MSK_IPAR_MIO_NODE_SELECTION (*integer)*

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

(default = 0)

0 The optimizer decides the node selection strategy.
1 The optimizer employs a depth first node selection strategy.
2 The optimizer employs a best bound node selection strategy.
3 The optimizer employs a worst bound node selection strategy.
4 The optimizer employs a hybrid strategy.
5 The optimizer employs selects the node based on a pseudo cost estimate.

MSK_IPAR_MIO_PRESOLVE_AGGREGATE (*integer)*

Controls whether the presolve used by the mixed integer optimizer tries to aggregate the constraints.

(default = 1)

MSK_IPAR_MIO_PRESOLVE_USE (*integer)*

Controls whether presolve is performed by the mixed integer optimizer.

(default = 1)

MSK_DPAR_MIO_REL_ADD_CUT_LIMITED (*real)*

Controls how many cuts the mixed integer optimizer is allowed to add to the problem. The mixed integer optimizer is allowed to `MIO_REL_ADD_CUT_LIMITED`$*m$ w cuts, where m is the number constraints in the problem.

Range: [0.0,2.0]

(default = 0.75)

MSK_IPAR_MIO_ROOT_OPTIMIZER (*integer)*

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

(default = 0)

0 The choice of optimizer is made automatically.
1 The interior-point optimizer is used.
2 Another cone optimizer.
3 The Qcone optimizer is used.
4 The primal simplex optimizer is used.
5 The dual simplex optimizer is used.
6 Either the primal or the dual simplex optimizer is used.
7 The mixed integer optimizer.

8 The optimizer for nonconvex nonlinear problems.

9 The concurrent optimizer is used.

MSK_IPAR_MIO_STRONG_BRANCH (*integer*)

The value specifies the depth from the root in which strong branching is used. A negative value means the optimizer chooses a default value automatically.

(default = -1)

MSK_DPAR_MIO_TOL_ABS_GAP (*real*)

Absolute optimality tolerance employed by the mixed integer optimizer.

(default = 0.0)

MSK_DPAR_MIO_TOL_ABS_RELAX_INT (*real*)

Absolute relaxation tolerance of the integer constraints, i.e. if the fractional part of a discrete variable is less than the tolerance, the integer restrictions assumed to be satisfied.

(default = 1.0e-5)

MSK_DPAR_MIO_TOL_REL_GAP (*real*)

Relative optimality tolerance employed by the mixed integer optimizer.

(default = 1.0e-8)

MSK_DPAR_MIO_TOL_REL_RELAX_INT (*real*)

Relative relaxation tolerance of the integer constraints, i.e. if the fractional part of a discrete variable is less than the tolerance times the level of that variable, the integer restrictions assumed to be satisfied.

(default = 1.0e-6)

MSK_IPAR_MIO_PRESOLVE_PROBING (*integer*)

Controls whether the mixed integer presolve performs probing. Probing can be very time consuming.

(default = 1)

MSK_IPAR_MIO_CUT_LEVEL_TREE (*integer*)

Controls the cut level employed by the mixed integer optimizer at the tree. See MSK_IPAR_MIO_CUT_LEVEL_ROOT.

(default = -1)

6 The MOSEK Log File

The MOSEK log output gives much useful information about the current solver progress and individual phases.

6.1 Log Using the Interior Point Optimizer

The following is a MOSEK log output from running the transportation model `trnsport.gms` from the GAMS Model Library:

```
Interior-point optimizer started.
Presolve started.
Linear dependency checker started.
Linear dependency checker terminated.
Presolve   - time                  : 0.00
Presolve   - Stk. size (kb)        : 0
Eliminator - tries                 : 0                 time                   : 0.00
Eliminator - elim's                : 0
```

```
Lin. dep.  - tries                  : 1                 time                    : 0.00
Lin. dep.  - number                 : 0
Presolve terminated.
Matrix reordering started.
Local matrix reordering started.
Local matrix reordering terminated.
Matrix reordering terminated.
Optimizer  - threads                : 1
Optimizer  - solved problem         : the primal
Optimizer  - constraints            : 5                 variables               : 11
Factor     - setup time             : 0.00              order time              : 0.00
Factor     - GP order used          : no                GP order time           : 0.00
Factor     - nonzeros before factor : 11                after factor            : 13
Factor     - offending columns      : 0                 flops                   : 2.60e+01
```

The first part gives information about the presolve (if used). The main log follows:

```
ITE PFEAS    DFEAS    KAP/TAU  POBJ              DOBJ              MU       TIME
0   6.0e+02  1.0e+00  1.0e+00  1.053000000e+00   0.000000000e+00   1.2e+01  0.00
1   5.9e+02  1.1e+00  1.0e+00  3.063646498e+00   5.682895191e+00   3.0e+01  0.00
2   4.6e+01  8.6e-02  9.8e+00  3.641071165e+01   4.750801284e+01   2.3e+00  0.00
3   8.7e-01  1.6e-03  1.7e+01  1.545771936e+02   1.719072826e+02   4.4e-02  0.00
4   8.1e-02  1.5e-04  8.8e-01  1.543678291e+02   1.552521470e+02   4.1e-03  0.00
5   1.3e-02  2.4e-05  1.3e-01  1.537617961e+02   1.538941635e+02   6.4e-04  0.00
6   1.3e-03  2.4e-06  1.1e-02  1.536766256e+02   1.536876562e+02   6.6e-05  0.00
7   1.6e-07  3.1e-10  1.2e-06  1.536750013e+02   1.536750025e+02   8.4e-09  0.00
Basis identification started.
Primal basis identification phase started.
ITER      TIME
1         0.00
Primal basis identification phase terminated. Time: 0.00
Dual basis identification phase started.
ITER      TIME
0         0.00
Dual basis identification phase terminated. Time: 0.00
Basis identification terminated. Time: 0.00
Interior-point optimizer terminated. CPU Time: 0.00. Real Time: 0.00.

Interior-point solution
Problem status  : PRIMAL_AND_DUAL_FEASIBLE
Solution status : OPTIMAL
Primal - objective: 1.5367500132e+02   eq. infeas.: 5.61e-06 max bound infeas.: 0.00e+00 cone infeas.:
Dual   - objective: 1.5367500249e+02   eq. infeas.: 1.06e-08 max bound infeas.: 0.00e+00 cone infeas.:

Basic solution
Problem status  : PRIMAL_AND_DUAL_FEASIBLE
Solution status : OPTIMAL
Primal - objective: 1.5367500000e+02   eq. infeas.: 0.00e+00 max bound infeas.: 0.00e+00
Dual   - objective: 1.5367500000e+02   eq. infeas.: 0.00e+00 max bound infeas.: 0.00e+00

Optimizer                  - real time  : 0.00       cpu time: 0.00
  Interior-point           - iterations : 7          cpu time: 0.00
    Basis identification   -                         cpu time: 0.00
      Primal               - iterations : 1          cpu time: 0.00
      Dual                 - iterations : 0          cpu time: 0.00
      Clean                - iterations : 0          cpu time: 0.00
  Simplex                -                          cpu time: 0.00
    Primal simplex       - iterations : 0           cpu time: 0.00
    Dual simplex         - iterations : 0           cpu time: 0.00
  Mixed integer          - relaxations: 0           cpu time: 0.00
```

The last section gives details about the model and solver status, primal and dual feasibilities, as well as solver resource times. Furthermore, the log gives information about the basis identification phase. Some of this information is listed in the GAMS solve summary in the model listing (.LST) file as well.

The fields in the main MOSEK log output are:

Field	Description
ITE	The number of the current iteration.
PFEAS	Primal feasibility.
DFEAS	Dual feasibility.
KAP/TAU	This measure should converge to zero if the problem has a primal/dual optimal solution. Whereas it should converge to infinity when the problem is (strictly) primal or dual infeasible. In the case the measure is converging towards a positive but bounded constant then the problem is usually ill-posed.
POBJ	Current objective function value of primal problem.
DOBJ	Current objective function value of dual problem.
MU	Relative complementary gap.
TIME	Current elapsed resource time in seconds.

6.2 Log Using the Simplex Optimizer

Below is a log output running the model `trnsport.gms` from the GAMS model library using the MOSEK simplex optimizer.

```
Reading parameter(s) from "mosek.opt"
>>  MSK_IPAR_OPTIMIZER 6
Simplex optimizer started.
Presolve started.
Linear dependency checker started.
Linear dependency checker terminated.
Presolve   - time                  : 0.00
Presolve   - Stk. size (kb)        : 0
Eliminator - tries                 : 0                time                   : 0.00
Eliminator - elim's                : 0
Lin. dep.  - tries                 : 1                time                   : 0.00
Lin. dep.  - number                : 0
Presolve terminated.
Dual simplex optimizer started.
Dual simplex optimizer setup started.
Dual simplex optimizer setup terminated.
Optimizer  - solved problem        : the primal
Optimizer  - constraints           : 5                variables              : 6
Optimizer  - hotstart              : no

ITER      DEGITER%  FEAS                  DOBJ                  TIME(s)
0         0.00      0.0000000000e+00      0.0000000000e+00      0.00
3         0.00      0.0000000000e+00      1.5367500000e+02      0.00
Dual simplex optimizer terminated.
Simplex optimizer terminated.
Basic solution
Problem status  : PRIMAL_AND_DUAL_FEASIBLE
Solution status : OPTIMAL
```

```
Primal - objective: 1.5367500000e+02   eq. infeas.: 0.00e+00 max bound infeas.: 0.00e+00
Dual   - objective: 1.5367500000e+02   eq. infeas.: 0.00e+00 max bound infeas.: 0.00e+00
```

The fields in the main MOSEK log output are:

Field	Description
ITER	Current number of iterations.
DEGITER%	Current percentage of degenerate iterations.
FEAS	Current (primal or dual) infeasibility.
D/POBJ	Current dual or primal objective
TIME	Current elapsed resource time in seconds.

6.3 Log Using the Mixed Integer Optimizer

Below is a log output running the model `cube.gms` from the GAMS model library using the MOSEK mixed-integer optimizer.

```
Mixed integer optimizer started.
BRANCHES RELAXS   ACT_NDS  BEST_INT_OBJ         BEST_RELAX_OBJ       REL_GAP(%)  TIME
0        1        0        1.6000000000e+01     0.0000000000e+00     100.00      0.1
0        1        0        4.0000000000e+00     0.0000000000e+00     100.00      0.1
128      250      5        4.0000000000e+00     0.0000000000e+00     100.00      0.3
167      502      6        4.0000000000e+00     0.0000000000e+00     100.00      0.7
241      758      65       4.0000000000e+00     0.0000000000e+00     100.00      0.9
200      809      83       4.0000000000e+00     1.3333333333e-01     96.67       0.9
A near optimal solution satisfying the absolute gap tolerance of 3.90e+00 has been located.


Objective of best integer solution : 4.00000000e+00
Number of branches                 : 267
Number of relaxations solved       : 810
Number of interior point iterations: 0
Number of simplex iterations       : 10521
Mixed integer optimizer terminated. Time: 0.95
```

The fields in the main MOSEK log output are:

Field	Description
BRANCHES	Current number of branches in tree.
RELAXS	Current number of nodes in branch and bound tree.
ACT_NDS	Current number of active nodes.
BEST_INT_OBJ.	Current best integer solution
BEST_RELAX_OBJ	Current best relaxed solution.
REL_GAP(%)	Relative gap between current BEST_INT_OBJ. and BEST_RELAX_OBJ.
TIME	Current elapsed resource time in seconds.

The log then gives information about solving the model with discrete variables fixed in order to determine marginals. We also get information about crossover to determine a basic solution, and finally MOSEK provides information about using the Simplex Method to determine an optimal basic solution.

```
Interior-point optimizer started.
Presolve started.
Linear dependency checker started.
Linear dependency checker terminated.
```

```
Presolve   - time                   : 0.00
Presolve   - Stk. size (kb)         : 12
Eliminator - tries                  : 0                 time                   : 0.00
Eliminator - elim's                 : 0
Lin. dep.  - tries                  : 1                 time                   : 0.00
Lin. dep.  - number                 : 0
Presolve terminated.
Interior-point optimizer terminated. CPU Time: 0.00. Real Time: 0.00.

Interior-point solution
Problem status  : PRIMAL_FEASIBLE
Solution status : PRIMAL_FEASIBLE
Primal - objective: 4.0000000000e+00   eq. infeas.: 0.00e+00 max bound infeas.: 0.00e+00 cone infeas.:
Dual   - objective: -8.0000000000e+00  eq. infeas.: 0.00e+00 max bound infeas.: 0.00e+00 cone infeas.:

Basic solution
Problem status  : PRIMAL_FEASIBLE
Solution status : PRIMAL_FEASIBLE
Primal - objective: 4.0000000000e+00   eq. infeas.: 0.00e+00 max bound infeas.: 0.00e+00
Dual   - objective: -8.0000000000e+00  eq. infeas.: 0.00e+00 max bound infeas.: 0.00e+00

Optimizer                - real time  : 1.35       cpu time: 0.95
  Interior-point         - iterations : 0          cpu time: 0.00
    Basis identification -                         cpu time: 0.00
      Primal             - iterations : 0          cpu time: 0.00
      Dual               - iterations : 0          cpu time: 0.00
      Clean              - iterations : 0          cpu time: 0.00
  Simplex                -                         cpu time: 0.00
    Primal simplex       - iterations : 0          cpu time: 0.00
    Dual simplex         - iterations : 0          cpu time: 0.00
  Mixed integer          - relaxations: 810        cpu time: 0.95
```