MOSEK version 6.
MOSEK Technical report: TR-2009-3.

Erling D. Andersen Bo Jensen Jens Jensen Rune Sandvik
Ulf Worsøe [*]

October 20, 2009

Abstract

This report presents the major new features in MOSEK version 6 and benchmark results. The benchmark results demonstrates the simplex optimizers reduce the solution time 25% on average compared to version 5. Furthermore, the new interior-point optimizer reduces the solution time by about 10% on average. However for conic problems a reduction 25% in the solution time is obtained.

1 Introduction

The MOSEK optimization tools software package is intended at solution of large-scale optimization problems of the following form

- linear,
- convex quadratic,
- conic,
- convex nonlinear, and
- mixed-integer

Moreover, MOSEK optimization tools include interfaces that make it easy to deploy the functionality of MOSEK from programming environments such as C, C++, Java, MATLAB, .NET, and Python.

The major new improvements and features in MOSEK version 6 are:

- Improved speed and stability of the interior-point and simplex optimizers.
- An improved Python interface.
- A problem analyzer that is useful for debugging purposes.
- Support for MATLAB R2009b.

Subsequently we will discuss in some detail the improvements in details

[*]MOSEK ApS, Fruebjergvej 3 Box 16, 2100 Copenhagen, Denmark

2 Improvements in the optimizers

2.1 The simplex optimizers

The main focus areas for the simplex optimizers are:

- Speed. On average the simplex optimizers are approximately 25% faster than the version 5 implementation.
  - The hybrid pricing in primal simplex optimizer has been redesigned.
  - The automatic switch between partial and approximate steepest-edge pricing in primal simplex has been improved.
  - Restricted pricing (i.e optimizing over a subset of variables) is now used more intensively.
  - Better crash module in dual simplex.
- Robustness. The numerical stability has been greatly improved.

2.2 The interior-point optimizer

The linear and conic interior-point optimizer in MOSEK have been improved. The most important improvements are:

- Speed. The linear interior-point optimizer is on average 10% faster for large scale linear problems. For conic problems the improvement is usually bigger.
- Numerical stability. The numerical stability of the linear and conic interior-point optimizers have been improved.
- Stopping criterion. The stopping criterion has been improved and documented in the manual.

2.3 The Python interface

The Python interface has been significantly improved. The interface now uses the Python CTypes module, which means that it should be compatible with all Python 2.x distributions that include the CTypes module. Currently this means Python 2.5 and later for most platforms. Python 2.4 is supported, but this requires that the CTypes module is installed separately.

In MOSEK version 6 NumPy is now the preferred array package compared to the older Numeric package in previous MOSEK versions. If NumPy is not available, it is still possible to use the MOSEK array module instead.

The new implementation improves the Python API in general, but causes some incompatibilities with code written for MOSEK 5.0. Mainly:

- The `msk` object no longer exists; the environment module is created without it.
- The `mosekarr` module is now accessed as `mosek.array`.
- Functions may return multiple values instead of taking a `list` argument. For example

  ```
  Task.getsolution()
  ```

  now returns `(prosta,solsta)` as a tuple.

2.4 The interactive shell

An interactive shell based on the Python interface has been added on the Windows platform. It can be invoked from start menu or from the command line as follows

```
moseki
```

2.5 The problem analyzer

In MOSEK version 6 a problem analyzer has been added. The purpose of the analyzer is to report statistics about problem that might useful in a debugging context. An example report is shown below:

```
Analyzing the problem

Constraints              Bounds                   Variables
 upper bd:       421      ranged  : all            cont:       421
 fixed   :        58                               bin :       421

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

Objective, min cx
   range: min |c|: 0.00000   min |c|>0: 11.0000     max |c|: 500.000
 distrib:        |c|        vars
                   0         421
           [11, 100)         150
          [100, 500]         271

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

Constraint matrix A has
       479 rows (constraints)
       842 columns (variables)
      2091 (0.518449%) nonzero entries (coefficients)

Row nonzeros, A_i
   range: min A_i: 2 (0.23753%)    max A_i: 34 (4.038%)
 distrib:        A_i        rows       rows%        acc%
                   2         421       87.89       87.89
             [8, 15]          20        4.18       92.07
            [16, 31]          30        6.26       98.33
            [32, 34]           8        1.67      100.00

Column nonzeros, A|j
   range: min A|j: 2 (0.417537%)    max A|j: 3 (0.626305%)
 distrib:        A|j        cols       cols%        acc%
                   2         435       51.66       51.66
                   3         407       48.34      100.00

A nonzeros, A(ij)
   range: min |A(ij)|: 1.00000     max |A(ij)|: 100.000
 distrib:      A(ij)      coeffs
             [1, 10)        1670
           [10, 100]         421

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

Constraint bounds, lb <= Ax <= ub
 distrib:        |b|             lbs             ubs
```

```
                  0                             421
            [1, 10]             58               58

Variable bounds, lb <= x <= ub
 distrib:        |b|            lbs             ubs
                  0             842
            [1, 10)                             421
          [10, 100]                             421

------------------------------------------------------------------------------
```

3 Computational results

This section will demonstrate the capabilities of the optimizers in MOSEK 6.

The computational results will be presented using a number of tables that usually has the format of the example Table 1. Subsequently we will explain Table 1 in details.

Table 1 shows a fictive run comparing version 5 and version 6 of an optimizer. Hence, a number of test problems has been solved using version 5 and version 6 of an optimizer e.g. the primal simplex optimizer. We categorize the test problems into small, medium, and large sized problems. The class of small problems consist of all those problems that the fastest optimizer can solve in less than 6 seconds. Similarly the class of medium sized problems consist of all those problems where the fastest optimizer require at least 6 seconds and no more than 60 seconds. Problems that where solved successfully by all the optimizers within the time limited and requiring more than 60 are denoted large problems. The time limited per instance per optimizer is 3600 seconds unless otherwise stated explicit. If a problem is not solved by one of the optimizers then it is exclude from statistics and included in the failure count for the optimizer.

In Table 1 we report how many problems that where solved by each optimizer in the row named "Num." for each class of problems. Next we report how many times an optimizer is first in the row "Firsts". More precisely we say an optimizer is first on a problem if the optimizer solve a problem within a time that is at most 1% slower than the fastest optimizer. Let t_5 and t_6 be the time that the version 5 and version 6 optimizer require to solve a problem instance then define the ratio r as follows

$$r := \frac{t_6 + 0.01}{t_5 + 0.01}.$$

"Geo. avg. r" shows the geometric average of all ratios for each instance.

From Table 1 it is seen that version 6 solved three out of four problems faster than version 5 in the group of small problems. Moreover, for the small problems version 6 and version 5 requires 40 seconds and 80 seconds respectively to solve all the small problems.

	small		medium		large		failures	
	6	5	6	5	6	5	6	5
Num	4	4	20	20	13	13	0	1
Firsts	3	1	15	5	10	3		
Total time	40	80	210	703	3789	10456		
G. avg. r	0.72		0.62		0.75			

Table 1: Example table.

As time measurements we use the optimizer time reported by MOSEK, which excludes time reading and writing files. It should be emphasized that times in MOSEK version 5 is CPU time whereas in MOSEK version 6 it is (high resolution) real clock time. This implies the comparison for small problems taking less than 1 second is not very reliable.

In order to obtain the test results we employ public available test problems such as the NETLIB and the Kennington problems as well as problems collected from our users. Finally, all results are obtained using the following computer:

- Intel Xeon 2.27 GHz 16 Cores.
- 16 GB RAM.
- Running Linux.

3.1 The simplex optimizers

As mentioned in Section 2.1 the simplex optimizers have been improved. In the following sections we will show test results for the version 5 and 6 optimizers on a number of test problems.

3.1.1 The primal simplex optimizer

In Table 2 the performances of the version 5 and version 6 primal simplex optimizer are compared. Independent of the problem size the version 6 optimizer has the most "firsts" and the lowest total time. For the large problems the solution time is approximately reduced by 25%. It should be emphasized that due change from using CPU time in version to real clock time in version 6 then the results for the group of small problems is not very reliable.

	small		medium		large	
	6	5	6	5	6	5
Num.	697	697	83	83	45	45
Firsts	345	518	56	28	32	13
Total time	676.67	1162.88	2108.09	3341.62	33620.24	40277.05
G. avg. r	1.23		0.76		0.75	

Table 2: Performance of the version 5 and version 6 primal simplex optimizer.

3.1.2 The dual simplex optimizer

In Table 3 the performance of the version 5 and version 6 dual simplex optimizer is compared. The version 6 optimizer is comparable to version 5 optimizer on the small sized problems. For medium and large sized problems the average time is reduced by approximately 25%.

	small		medium		large	
	6	5	6	5	6	5
Num.	723	723	68	68	42	42
Firsts	478	489	42	26	24	19
Total time	529.54	863.86	1759.39	3395.73	23869.25	36741.48
G. avg. r	1.14		0.75		0.75	

Table 3: Performance of the version 5 and version 6 dual simplex optimizer

To summarize, the stability and speed of the simplex optimizers has been improved significantly in version 6 compared to version 5.

3.2 The interior-point optimizer

The interior-point optimizers for linear and conic problems has been improved. Notably we have

- cleaned up the implementation,
- improved the robustness,
- improved the infeasibility handling, and
- improved the termination criteria.

3.2.1 Linear problems

Table 4 shows results of the interior-point optimizer applied to a test set of approximately 1500 linear optimization problems. For both small, medium, and large sized problems the version 6 optimizer has more wins than the version 5 optimizer. Moreover, for medium and large sized problems version 6 is 5 to 10% faster than version 5.

	small		medium		large		failures	
	6	5	6	5	6	5	6	5
Num.	1348	1348	141	141	63	63	4	7
Firsts	1048	801	100	43	42	22		
Total time	750	918	3200	5967	15980	15469		
G. avg. r	1.14		0.88		0.95			

Table 4: Performance on the version 5 and 6 of the interior-point optimizer.

3.2.2 Conic problems

Table 5 present results of the interior-point optimizer when applied to a test set of approximately 200 conic optimization problems. The table documents that for the majority of the problems version 6 is faster than version 5. On average the solution time is improved by at least 10% and for the (few) large problems the solution is improved by 20%. Furthermore, we have reduced the number of failures by approximately 30%. The number failures is a bit too larger than we would like. However, it should observed that the conic test problems are likely to be harder than the “average” conic optimization since we usually only obtain problems from customers when they causing an issue in MOSEK.

	small		medium		large		failures	
	6	5	6	5	6	5	6	5
Num.	175	175	29	29	10	10	24	35
Firsts	120	95	23	6	8	2		
Total time	112	307	570	2411	1368	2005		
G. avg. r	0.96		0.66		0.75			

Table 5: Performance on the version 5 and 6 of the interior-point optimizer.

4 Conclusion

This report has discussed the performance improvements gained from version 5 to version 6. For medium to large sized problems the primal simplex, the dual simplex and the interior-point optimizers are approximately 25%, 25% and 10% faster respectively in version 6. Moreover, for conic problems the improvement in the conic optimizer is about 25% faster for medium to large sized problems.