# The Codes

The submitted codes fall naturally into two major groups and several subgroups:

* The first major group is that of primal-dual interior point methods designed for small to medium sized problems.
  + In the first subgroup are the codes SeDuMi, and SDPT3. These codes handle all 3 types of cones. SDPT3 was enabled to handle second order cones during the course of the Challenge.
  + In the second subgroup are SDPA, and CSDP, which are limited to SDP.
  + In the third subgroup are codes not designed for SDP problems but for convex (MOSEK) and nonconvex as well as convex (LOQO) nonlinear optimization. In fact, late in the Challenge an SDP interface for LOQO was provided but it did not solve satisfactorily any of the Challenge problems and these results were not included below. LOQO did solve some smaller SDPLIB problems.
* The second group is that of large-scale SDP codes designed to provide approximate solutions for large scale problems: BMPR, BMZ, BUNDLE and DSDP. The first three of these do not make use of second order derivative information, while DSDP is a dual interior point code.

In the following the codes will be listed in the above order.

The input formats are:

**graph**: Graph format as provided by the organizers.

**Matlab**: SeDuMi format in Matlab binary form as provided by organizers.

**QPS**: extended MPS format as explained in the MOSEK user's guide [[20](http://plato.asu.edu/dimacs/node16.html#MOSEKGUIDE)]; this was generated from the Matlab format with a converter provided by E. Andersen.

**SDPA**: the sparse SDPA format as explained in the SDPA user's guide; problems not provided in this format were converted from Matlab format to SDP formulation with the help of a program provided by B. Borchers.

**SDPA**

**Authors:** Fujisawa, Kojima, Nakata   
**Version** 5.02, 9/2000; **Available:** yes; the software manual and the SDPA source codes can be found at http://www.is.titech.ac.jp/ yamashi9/sdpa/index.html   
**Key paper:** [[11](http://plato.asu.edu/dimacs/node16.html#FUJISAWA00)]. For implementation and numerical results - [[9](http://plato.asu.edu/dimacs/node16.html#FUJISAWA99)].   
**Features:** primal-dual method, tested version uses Meschach library   
**Language, Input format:** C++; SDPA   
**Error computations:** yes   
**Solves:** SDP

The SDPA (SemiDefinite Programming Algorithm) is based on a Mehrotra-type predictor-corrector infeasible primal-dual interior-point method [[17](http://plato.asu.edu/dimacs/node16.html#KOJIMA97),[9](http://plato.asu.edu/dimacs/node16.html" \l "FUJISAWA99)], and is implemented in the C++ language utilizing the *Meschach* [[23](http://plato.asu.edu/dimacs/node16.html#STEWART94)] library for matrix computations. The main features are: it is available in a callable library, three types of search directions can be used (H..K..M, NT and AHO), block diagonal and sparse data matrix structures are exploited, and information on infeasibility is provided. SDPA uses a set of parameters which the user can adjust to cope with numerically difficult semidefinite programs.

Stopping criteria:

* $ \min \{ \left \Vert\mathcal{A}x - b\Vert _\infty,\Vert\mathcal{A}^* y + z - c\Vert _\infty \right \} \leq \epsilon'$and
* $ \frac{\vert\langle c, x \rangle - b^T y\vert}{\max \left \{(\vert\langle c, x \rangle \vert + \vert b^T y\vert)\vert/2.0, 1.0 \right \} } \leq \epsilon^*$

Typical values of the parameters $ \epsilon'$and $ \epsilon^*$are $ 10^{-6} \sim 10^{-8}$. See [[9](http://plato.asu.edu/dimacs/node16.html#FUJISAWA99),[11](http://plato.asu.edu/dimacs/node16.html" \l "FUJISAWA00)] for more details.

**SeDuMi**

**Author:** Sturm   
**Version:** 1.04, 9/2000;   
**Available:** yes, from http://fewcal.kub.nl/sturm/software/sedumi.html   
**Key papers:** [[24](http://plato.asu.edu/dimacs/node16.html#S98guide),[25](http://plato.asu.edu/dimacs/node16.html" \l "S00CH7)]   
**Features:** self-dual embedding, dense column handling   
**Language, Input format:** Matlab+C; Matlab, SDPA, SDPpack   
**Error computations:** yes   
**Solves:** SDP/SOCP

The primal-dual interior point algorithm implemented in SeDuMi [[24](http://plato.asu.edu/dimacs/node16.html#S98guide)] is described in [[25](http://plato.asu.edu/dimacs/node16.html#S00CH7)]. The algorithm has an $ O(\sqrt{n} \log \epsilon)$worst case bound, and treats initialization issues by means of the self-dual embedding technique of [[28](http://plato.asu.edu/dimacs/node16.html#YTM94sdual)]. The iterative solutions are updated in a product form, which makes it possible to provide highly accurate solutions.

The algorithm terminates successfully if the norm of the residuals in the optimality conditions, or the Farkas system with $ b^T y=1$or $ \langle c, x \rangle = -1$, are less than the parameter pars.eps. The default value for pars.eps is 1E-9.

**Remarks:**

* SeDuMi exploits sparsity in solving the normal equations; this results in a benefit for problems with a large number of small order matrix variables, such as the *copositivity*-problems in the Dimacs set.
* However, for problems that involve a huge matrix variable (without a block diagonal structure), the implementation is slow and consumes an excessive amount of memory.

**SDPT3**

**Authors:** Toh, Todd, Tütüncü   
**Version:** 3.0, 7/2001;   
**Available:** yes, from http://www.math.nus.edu.sg/ mattohkc/sdpt3.html   
**Key paper:** [[26](http://plato.asu.edu/dimacs/node16.html#SDPT3)]   
**Features:** primal-dual method, infeasible primal-dual and homogeneous self-dual formulations, Lanczos steplength computation   
**Language, Input format:** Matlab+C or Fortran; SDPA   
**Error computations:** yes   
**Solves:** SDP and SOCP

This code is designed to solve conic programming problems whose constraint cone is a product of semidefinite cones, second-order cones, and/or nonnegative orthants. It employs a predictor-corrector primal-dual path-following method, with either the H..K..M or the NT search direction. The basic code is written in Matlab, but key subroutines in Fortran and C are incorporated via Mex files. Routines are provided to read in problems in either SeDuMi or SDPA format. Sparsity and block diagonal structure are exploited, but the latter needs to be given explicitly.

The algorithm is stopped if:

* primal infeasibility is suggested because      $ b^T y / \Vert \mathcal{A}^* y + z \Vert > 10^8;$or
* dual infeasibility is suggested because      $ - \langle c, x \rangle / \Vert\mathcal{A}x \Vert > 10^8;$or
* sufficiently accurate solutions have been obtained:

\begin{displaymath}
\mbox{rel\_gap} := \frac{\langle x, z \rangle }{\max\{1,(\langle c, x \rangle + b^T y)/2\}}
\end{displaymath}

and

\begin{displaymath}
\mbox{infeas\_meas} := \max \left[
\frac{\Vert\mathcal{A}x ...
...t\mathcal{A}^* y + z - c\Vert}{\max\{1,\Vert c\Vert\}} \right]
\end{displaymath}

are both below $ 10^{-8}$; or

* slow progress is detected, measured by a rather complicated set of tests including

\begin{displaymath}
\langle x, z \rangle / n < 10^{-4} \quad \mbox{and} \quad
\mbox{rel\_gap} < 5 * \mbox{infeas\_meas};
\end{displaymath}

or

* numerical problems are encountered, such as the iterates not being positive definite, the Schur complement matrix not being positive definite, or the step sizes falling below $ 10^{-6}$.

**Remarks:**

* SDPT3 is a general-purpose code based on a polynomial-time interior-point method.
* It should obtain reasonably accurate solutions to problems of small and medium size (for problems with semidefinite constraints, up to around a thousand constraints involving matrices of order up to around a thousand, and for sparse problems with only second-order/linear cones, up to around 20,000 constraints and 50,000 variables), and can solve some larger problems.
* Because it uses a primal-dual strategy, forms the Schur complement matrix for the Newton equations, and employs direct methods, it is unlikely to compete favorably with alternative methods on large-scale problems.