

Research Reports on Mathematical and Computing Sciences

SDPA (SemiDefinite Programming Algorithm)
User's Manual — Version 7.0.5

Katsuki Fujisawa, Mituhiro Fukuda, Kazuhiro
Kobayashi, Masakazu Kojima, Kazuhide Nakata,
Maho Nakata, and Makoto Yamashita

February 2008, B-448

Department of
Mathematical and
Computing Sciences
Tokyo Institute of Technology

SERIES **B: Operations Research**

SDPA (SemiDefinite Programming Algorithm)
User's Manual — Version 7.0.5

Katsuki Fujisawa^{*1}, Mituhiro Fukuda^{*2}, Kazuhiro Kobayashi^{*3}, Masakazu Kojima^{*4},
Kazuhide Nakata^{*5}, Maho Nakata^{*6}, and Makoto Yamashita^{*7}

Abstract. The SDPA (SemiDefinite Programming Algorithm) [5] is a software package for solving semidefinite programs (SDPs). It is based on a Mehrotra-type predictor-corrector infeasible primal-dual interior-point method. The SDPA handles the standard form SDP and its dual. It is implemented in C++ language utilizing the *LAPACK* [1] for matrix computations. The SDPA version 7.0.5 enjoys the following features:

- Efficient method for computing the search directions when the SDP to be solved is large scale and sparse [4].
- Block diagonal matrix structure and sparse matrix structure are supported for data matrices.
- Sparse or dense Cholesky factorization for the Schur matrix is automatically selected.
- An initial point can be specified.
- Some information on infeasibility of the SDP is provided.

This manual and the SDPA can be downloaded from the WWW site
<http://sdpa.indsys.chuo-u.ac.jp/sdpa/index.html>

Key words. Semidefinite programming, interior-point method, computer software.

- *1 Department of Industrial and Systems Engineering
Chuo University
1-13-27 Kasuga, Bunkyo-ku, Tokyo 112-8551, Japan
- *2 Global Edge Institute
Tokyo Institute of Technology
2-12-1-S6-5 Oh-Okayama, Meguro-ku, Tokyo 152-8550, Japan
- *3 Center for Logistics Research
National Maritime Research Institute
6-38-1 Shinkawa, Mitaka-shi, Tokyo 181-0004, Japan
- *4 Department of Mathematical and Computing Sciences
Tokyo Institute of Technology
2-12-1-W8-29 Oh-Okayama, Meguro-ku, Tokyo 152-8552, Japan
Corresponding Author: kojima@is.titech.ac.jp
- *5 Department of Industrial Engineering and Management
Tokyo Institute of Technology
2-12-1-W9-62, Oh-Okayama, Meguro-ku, Tokyo 152-8552, Japan
- *6 Advanced Center for Computing and Communication
RIKEN
2-1 Hirosawa, Wako-shi, Saitama 351-0198, Japan
- *7 Department of Information Systems Creation
Kanagawa University
3-27-1 Rokkakubashi, Kanagawa-ku, Yokohama-shi, Kanagawa 221-8686, Japan

Preface

We are pleased to release a new version 7.0.5 of the SDPA.

The SDPA 7.0.5 was completely revised from its source code, and there are great performance improvements on its computational time and memory usage. In particular, it uses and stores less variables internally, and its overall memory usage is less than half of the previous version. Also it performs sparse Cholesky factorization when the Schur Complement Matrix is sparse. As a consequence, the SDPA can efficiently solve SDPs with a large number of block diagonal matrices or non-negative constraints. Additional, there are some improvements on its numerical stability due to a better control in the interior-point algorithm.

The differences between this version and the previous version, 6.2.1, are summarized in Section 9.

We hope that the SDPA supports many researches in various fields. We also welcome any suggestions and comments that you may have. When you want to contact us, please send an e-mail to the following address.

`kojima-sdpa@is.titech.ac.jp`

Contents

1. Build and Installation	1
1.1 Prerequisites	1
1.2 How to Obtain the Source Code	1
1.3 How to Build	2
1.3.1 Fedora Core 6 (i386/x86_64), Fedora 7 and Fedora 8 (i386/x86_64/ppc), Red Hat Enterprise Linux (CentOS) 4, 5 (i386/x86_64)	2
1.3.2 Ubuntu 7.10 Desktop (i386/x86_64)	2
1.3.3 Vine Linux 4.2 (i386)	2
1.3.4 MacOSX Leopard (Intel/PowerPC)	3
1.3.5 MacOSX Tiger (Intel/PowerPC)	3
1.3.6 FreeBSD 6 and 7	3
1.4 How to Install	4
1.5 Test Run	4
1.6 Performance Tuning: Optimized BLAS and LAPACK	5
1.6.1 Configure Options	5
1.6.2 Linking Against ATLAS	6
1.6.3 Linking Against GotoBLAS	6
1.6.4 Linking Against Intel Math Kernel Library	7
1.7 Performance Tuning: Compiler Options	7
2. Semidefinite Program	7
2.1 Standard Form SDP and Its Dual	7
2.2 Example 1	9
2.3 Example 2	9
3. Files Necessary to Execute the SDPA	10
4. Input Data File	10
4.1 “example1.dat” — Input Data File of Example 1	10
4.2 “example2.dat” — Input Data File of Example 2	10
4.3 Format of the Input Data File	11
4.4 Title and Comments	11
4.5 Number of the Primal Variables	12
4.6 Number of Blocks and the Block Structure Vector	12
4.7 Constant Vector	13
4.8 Constraint Matrices	13

5. Parameter File	15
6. Output	16
6.1 Execution of the SDPA	16
6.2 Output on the Display	16
6.3 Output to a File	19
6.4 Printing DIMACS Errors	21
7. Advanced Use of the SDPA	22
7.1 Initial Point	22
7.2 Sparse Input Data File	23
7.3 Sparse Initial Point File	23
7.4 Obtaining More Precision on the Approximate Solution	24
7.5 More on Parameter File	25
8. Transformation to the Standard Form of SDP	25
8.1 Inequality Constraints	25
8.2 Norm Minimization Problem	27
8.3 Linear Matrix Inequality (LMI)	27
8.4 SDP Relaxation of the Maximum Cut Problem	28
8.5 Choosing Between the Primal and Dual Standard Forms	28
9. For the SDPA 6.2.1 Users	29

1. Build and Installation

This section describes how to build and install the SDPA. Usually, the SDPA is distributed as source codes, therefore, users must build it by themselves. We have done build and installation tests on Fedora Core 6 (i386/x86_64), Fedora 7, 8 (i386/x86_64/ppc), Red Hat Enterprise Linux (CentOS) 4, 5 (i386/x86_64), Ubuntu 7.10 (i386/x86_64), Vine Linux 4.2 (i386), MacOSX Leopard/Tiger (Intel/PowerPC) and FreeBSD 6/7. You may also possibly build on other UNIX like platforms and the Windows cygwin environment.

For better performance, the SDPA should link against optimized Basic Linear Algebra Subprograms (BLAS) and Linear Algebra PACKage (LAPACK). Please refer to the Section 1.6 for more details.

Alternatively, you can also use our online solver, or check for pre-build binary packages at the SDPA homepage (see Section 1.2).

1.1 Prerequisites

It is necessary to have at least the following programs installed on your system except for the MacOSX.

- C, C++ compiler.
- FORTRAN compiler.
- BLAS (<http://www.netlib.org/blas/>) and LAPACK (<http://www.netlib.org/lapack/>).

For MacOSX Leopard (Intel/PowerPC), you will need the Xcode 3.0, and you cannot build the SDPA on the Leopard with Xcode 2.5. For MacOSX Tiger (Intel/PowerPC), you will need either the Xcode 2.4.1 or the Xcode 2.5.

You can download the Xcode at Apple Developer Connection (<http://developer.apple.com/>) at free of charge.

In the following instructions, we install C, C++, FORTRAN compilers, BLAS and LAPACK and/or Xcode as well. You can skip this part if your system already have them. If not, you may need root access or administrative privilege to your computer. Please ask your system administrator for installing development tools.

If your system only lacks BLAS and LAPACK, you can build them by yourself without root privileges and passing appropriate flags at the command line.

1.2 How to Obtain the Source Code

You can obtain the source code following the links at the SDPA homepage:
<http://sdpa.indsys.chuo-u.ac.jp/sdpa/index.html>

You can find the latest news at about the SDPA at the [index.html](#) file.

1.3 How to Build

This section describes how to build the SDPA on Fedora Core 6 (i386/x86_64), Fedora 7, 8 (i386/x86_64/ppc), Ubuntu 7.10 (i386/x86_64), Vine Linux 4.2 (i386) MacOSX Intel/PowerPC Leopard (Xcode 3.0)/Tiger (Xcode 2.4.1 and 2.5) and FreeBSD 6/7. We assume that users have a root access via the command “su” or “sudo”. If you do not have such privileges, please ask your system administrators.

1.3.1 Fedora Core 6 (i386/x86_64), Fedora 7 and Fedora 8 (i386/x86_64/ppc), Red Hat Enterprise Linux (CentOS) 4, 5 (i386/x86_64)

To build the SDPA, type the following.

```
$ bash
$ su
Password:
# yum update glibc glibc-common
# yum install gcc gcc-c++ gcc-gfortran
# yum install lapack lapack-devel blas blas-devel
# yum install atlas atlas-devel
# exit
$ tar xvfz sdpa-7.0.5.tar.gz
$ cd sdpa-7.0.5
$ ./configure
$ make
```

1.3.2 Ubuntu 7.10 Desktop (i386/x86_64)

To build the SDPA, type the following.

```
$ bash
$ sudo apt-get install g++ patch
$ sudo apt-get install lapack3 lapack3-dev
$ sudo apt-get install atlas3-base atlas3-base-dev
$ tar xvfz sdpa-7.0.5.tar.gz
$ cd sdpa-7.0.5
$ ./configure
$ make
```

1.3.3 Vine Linux 4.2 (i386)

```
$ bash
$ su
```

Modify /etc/apt/sources.list (add the word “extras” between “updates” and “nonfree”) from

```
rpm      [vine] http://updates.vinelinux.org/apt 4.2/$(ARCH) main plus updates nonfree
rpm-src  [vine] http://updates.vinelinux.org/apt 4.2/$(ARCH) main plus updates nonfree
```

to

```
rpm [vine] http://updates.vinelinux.org/apt 4.2/$(ARCH) main plus updates extras nonfree
rpm-src [vine] http://updates.vinelinux.org/apt 4.2/$(ARCH) main plus updates extras nonfree
```

Then, type

```
# apt-get update
# apt-get install gcc-g77
# apt-get install lapack lapack-devel blas blas-devel
# exit
$ tar xvfz sdpa-7.0.5.tar.gz
$ cd sdpa-7.0.5
$ ./configure
$ make
```

1.3.4 MacOSX Leopard (Intel/PowerPC)

Download and install the Xcode 3 from Apple Developer Connection (<http://developer.apple.com/>) and install it. Note that we support only the Xcode 3 on Leopard. We do not support Leopard with Xcode 2.5.

```
$ bash
$ tar xvfz sdpa-7.0.5.tar.gz
$ cd sdpa-7.0.5
$ ./configure
$ make
```

1.3.5 MacOSX Tiger (Intel/PowerPC)

Download and install either the Xcode 2.4.1 or the Xcode 2.5 from Apple Developer Connection. Then type the following.

```
$ bash
$ tar xvfz sdpa-7.0.5.tar.gz
$ cd sdpa-7.0.5
$ ./configure
$ make
```

1.3.6 FreeBSD 6 and 7

```
$ cd /usr/ports
$ su
Password:
# make update
# cd /usr/ports/math/sdpa
# make install
# exit
```

1.4 How to Install

You will find the “sdpa” executable binary file at this point, and you can install it as:

```
$ su
Password:
# mkdir /usr/local/bin
# cp sdpa /usr/local/bin
# chmod 777 /usr/local/bin/sdpa
```

or you can install it to your favorite directory.

```
$ mkdir /home/nakata/bin
$ cp sdpa /home/nakata/bin
```

1.5 Test Run

Before using the SDPA, type “sdpa” and make sure that the following message will be displayed.

```
$ sdpa
SDPA start at    Tue Jan 22 15:28:59 2008

*** Please assign data file and output file.***

---- option type 1 -----
sdpa DataFile OutputFile [InitialPtFile] [-pt parameters]
parameters = 0 default, 1 aggressive, 2 stable
example1-1: sdpa example1.dat example1.result
example1-2: sdpa example1.dat-s example1.result
example1-3: sdpa example1.dat example1.result example1.ini
example1-4: sdpa example1.dat example1.result -pt 2

---- option type 2 -----
sdpa [option filename]+
  -dd : data dense  :: -ds : data sparse
  -id : init dense  :: -is : init sparse
  -o  : output      :: -p  : parameter
  -pt : parameters , 0 default, 1 aggressive
                        2 stable
example2-1: sdpa -o example1.result -dd example1.dat
example2-2: sdpa -ds example1.dat-s -o example2.result -p param.sdpa
example2-3: sdpa -ds example1.dat-s -o example3.result -pt 2
```

Let us solve the SDP “example1.dat-s”.

```
$ sdpa -ds example1.dat-s -o example1.out
SDPA start at    Fri Dec 7 18:30:56 2007
data           is example1.dat-s : sparse
parameter is  ./param.sdpa
out            is example1.out
```

```

DENSE computations
  mu      thetaP  thetaD  objP      objD      alphaP  alphaD  beta
0 1.0e+04 1.0e+00 1.0e+00 -0.00e+00 +1.20e+03 1.0e+00 9.1e-01 2.00e-01
1 1.6e+03 0.0e+00 9.4e-02 +8.39e+02 +7.51e+01 2.3e+00 9.6e-01 2.00e-01
2 1.7e+02 2.3e-16 3.6e-03 +1.96e+02 -3.74e+01 1.3e+00 1.0e+00 2.00e-01
3 1.8e+01 2.3e-16 1.5e-17 -6.84e+00 -4.19e+01 9.9e-01 9.9e-01 1.00e-01
4 1.9e+00 2.6e-16 1.5e-17 -3.81e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01
5 1.9e-01 2.6e-16 3.0e-17 -4.15e+01 -4.19e+01 1.0e-00 9.0e+01 1.00e-01
6 1.9e-02 2.5e-16 1.6e-15 -4.19e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01
7 1.9e-03 2.6e-16 2.2e-17 -4.19e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01
8 1.9e-04 2.5e-16 1.5e-17 -4.19e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01
9 1.9e-05 2.7e-16 7.5e-18 -4.19e+01 -4.19e+01 1.0e-00 9.0e+01 1.00e-01
10 1.9e-06 2.6e-16 5.0e-16 -4.19e+01 -4.19e+01 1.0e-00 9.0e+01 1.00e-01

```

```

phase.value = pdOPT
  Iteration = 10
    mu = 1.9180668442024010e-06
relative gap = 9.1554505577840628e-08
  gap = 3.8361336884048019e-06
  digits = 7.0383202783481345e+00
objValPrimal = -4.18999996163866390e+01
objValDual   = -4.1899999999999999e+01
p.feas.error = 3.2137639581876834e-14
d.feas.error = 4.7961634663806763e-13
total time   = 0.010
  main loop time = 0.010000
    total time = 0.010000
file  read time = 0.000000

```

1.6 Performance Tuning: Optimized BLAS and LAPACK

We highly recommend to use optimized BLAS and LAPACK for better performance, *e.g.*,

- Automatically Tuned Linear Algebra Software (ATLAS): <http://math-atlas.sourceforge.net/> . Note that in Section 1.3, we described how to use ATLAS which comes with Fedora Core 6, Fedora 7, 8, and Ubuntu. ATLAS optimizes itself while it is built, and we recommend users to rebuild it on the target computers. For convenience, some pre-build packages are available at : https://sourceforge.net/project/showfiles.php?group_id=23725 .
- GotoBLAS: <http://www.tacc.utexas.edu/resources/software/> .
- Intel Math Kernel Library (MKL): <http://www.intel.com/cd/software/products/asm-na/eng/266858.htm> .

Table 1 shows how the SDPA 7.0.5 performs on several benchmark problems when replacing the BLAS and LAPACK libraries. Typically, the SDPA 7.0.5 with optimized BLAS and LAPACK seems much faster than the one with BLAS/LAPACK 3.1.1. Furthermore, we can receive benefits of multi-thread computing when using SMP and/or multi-core machines.

1.6.1 Configure Options

There are some configure options you need to specify. We specify at least two of them when using optimized BLAS and LAPACK. Type "configure -help" for more details.

- --with-blas

Table 1: Numerical experiments (SDPA 7.0.5 + BLAS library). Time in seconds(# of iterations); CPU: Xeon 5345 (2.33GHz); OS: CentOS Ver 5 (64bit); Compiler: Intel (C/C++ & Fortran) 10.1.

BLAS library(# of threads)	Prob(1)	Prob(2)	Prob(3)	Prob(4)
BLAS/LAPACK 3.1.1(1)	72.32(35)	126.63(15)	221.31(18)	344.81(20)
ATLAS 3.8.0(1)	51.09(35)	49.73(15)	40.35(18)	180.64(19)
ATLAS 3.8.0(4)	30.06(35)	17.70(15)	17.09(18)	80.57(20)
Intel MKL 10.0.1.014(1)	44.97(36)	39.63(15)	34.27(18)	170.58(21)
Intel MKL 10.0.1.014(2)	31.44(34)	24.03(15)	22.23(18)	110.19(22)
Intel MKL 10.0.1.014(4)	26.18(35)	16.15(15)	16.17(18)	67.17(18)
Intel MKL 10.0.1.014(8)	24.06(35)	13.17(15)	14.18(18)	59.50(19)
GotoBLAS 1.22(1)	43.09(35)	40.23(15)	32.06(18)	157.18(21)
GotoBLAS 1.22(2)	32.16(35)	23.48(15)	21.15(18)	96.01(19)
GotoBLAS 1.22(4)	26.25(35)	15.11(15)	15.43(18)	68.77(19)
GotoBLAS 1.22(8)	24.66(36)	11.79(15)	13.67(18)	60.70(19)

Prob(1) : Structural Optimization
 Prob(2) : Combinatorial Optimization(1)
 Prob(3) : Combinatorial Optimization(2)
 Prob(4) : Quantum Chemistry

Supply a command line to link against your BLAS, e.g.,
`--with-blas="-L/home/nakata/gotoblas/lib -lgoto" .`

- `--with-lapack`

Supply a command line to link against your LAPACK, e.g.,
`--with-lapack="-L/home/nakata/gotoblas/lib -lgoto -llapack" .`

1.6.2 Linking Against ATLAS

Install ATLAS, and assume that the ATLAS libraries are installed at “/home/nakata/atlas/lib”. If “/home/nakata/atlas/lib/” is a full LAPACK archive, otherwise rename it. The ATLAS will not make a full LAPACK archive by default. Then, reconfigure and rebuild the SDPA like following.

```
$ make clean
$ ./configure --with-blas="-L/home/nakata/atlas/lib -lptf77blas -lptcblas -latlas"
$ make
```

The number of cores uses is determined at the build process, and you need a multi-core CPU to accelerate BLAS operations.

1.6.3 Linking Against GotoBLAS

Install GotoBLAS, and assume that the GotoBLAS libraries are installed at “/home/nakata/gotoblas/lib”. Reconfigure and rebuild the SDPA like following.

```
$ make clean
$ ./configure --with-blas="-L/home/nakata/gotoblas/lib -lgoto" \
  --with-lapack="-L/home/nakata/gotoblas/lib -lgoto -llapack"
$ make
```

You can set the number of threads uses via setting the “OMP_NUM_THREADS” environment variable before running the SDPA like following. In this case, two cores will be used for BLAS operations. You need a multi-core CPU to accelerate BLAS operations.

```
$ export OMP_NUM_THREADS=2
$ sdpa -ds example1.dat-s -o example1.out
```

1.6.4 Linking Against Intel Math Kernel Library

Install Intel Math Kernel Library, and assume that the library is installed at “/opt/intel/mkl/9.1.023/lib/em64t”. Reconfigure and rebuild the SDPA like following.

```
$ make clean
$ ./configure --with-blas="-L/opt/intel/mkl/9.1.023/lib/em64t -lmkl_lapack -lmkl_em64t" \
  --with-lapack="-L/opt/intel/mkl/9.1.023/lib/em64t -lmkl_lapack -lguide -lmkl_em64t"
$ make
```

You can set the number of threads uses via setting the “OMP_NUM_THREADS” environment variable before running the SDPA like following. In this case, four cores will be used for BLAS operations. You need a multi-core CPU to accelerate BLAS operations.

```
$ export OMP_NUM_THREADS=4
$ sdpa -ds example1.dat-s -o example1.out
```

1.7 Performance Tuning: Compiler Options

We also recommend adding some optimized flags to the compilers. Our recommendations are

- -O2 -funroll-all-loops (default)
- -static -O3 -funroll-all-loops -march=nocona -msse3 -mfpmath=sse (on Core2).

The above default options are sufficient for better performance in many cases. To pass your optimization flags, set them in CFLAGS and CXXFLAGS environment variables. Here is an example:

```
$ bash
$ make clean
$ export CFLAGS="-static -funroll-all-loops -O3 -m64 -march=nocona -msse3 -mfpmath=sse"
$ export CXXFLAGS="-static -funroll-all-loops -O3 -m64 -march=nocona -msse3 -mfpmath=sse"
$ ./configure --with-blas="-L/home/nakata/gotoblas/lib -lgoto" \
  --with-lapack="-L/home/nakata/gotoblas/lib -lgoto -llapack"
$ make
```

2. Semidefinite Program

2.1 Standard Form SDP and Its Dual

The SDPA (Semidefinite Programming Algorithm) solves the following standard form semidefinite program and its dual.

$$\text{SDP} \left\{ \begin{array}{ll} \mathcal{P}: & \text{minimize} \quad \sum_{i=1}^m c_i x_i \\ & \text{subject to} \quad \mathbf{X} = \sum_{i=1}^m \mathbf{F}_i x_i - \mathbf{F}_0, \quad \mathcal{S} \ni \mathbf{X} \succeq \mathbf{O}, \\ \mathcal{D}: & \text{maximize} \quad \mathbf{F}_0 \bullet \mathbf{Y} \\ & \text{subject to} \quad \mathbf{F}_i \bullet \mathbf{Y} = c_i \quad (i = 1, 2, \dots, m), \quad \mathcal{S} \ni \mathbf{Y} \succeq \mathbf{O}. \end{array} \right.$$

\mathcal{S} : set of $n \times n$ real symmetric matrices

$\mathbf{F}_i \in \mathcal{S}$ ($i = 0, 1, \dots, m$) : constraint matrices

$\mathbf{O} \in \mathcal{S}$: zero matrix

$$\mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{pmatrix} \in \mathbb{R}^m : \text{cost vector}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \in \mathbb{R}^m : \text{variable vector}$$

$\mathbf{X} \in \mathcal{S}, \mathbf{Y} \in \mathcal{S}$: variable matrices

$\mathbf{U} \bullet \mathbf{V}$: inner product between \mathbf{U} and $\mathbf{V} \in \mathcal{S}$, i.e., $\sum_{i=1}^n \sum_{j=1}^n U_{ij} V_{ij}$

$\mathbf{U} \succeq \mathbf{O} \iff \mathbf{U} \in \mathcal{S}$ is positive semidefinite

Throughout this manual, we denote the primal SDP by \mathcal{P} and its dual problem by \mathcal{D} . The SDP is determined by $m, n, \mathbf{c} \in \mathbb{R}^m$, and $\mathbf{F}_i \in \mathcal{S}$ ($i = 0, 1, \dots, m$). When (\mathbf{x}, \mathbf{X}) is a feasible solution (or a minimum solution, resp.) of the primal problem \mathcal{P} and \mathbf{Y} is a feasible solution (or a maximum solution, resp.), we call $(\mathbf{x}, \mathbf{X}, \mathbf{Y})$ a feasible solution (or an optimal solution, resp.) of the SDP.

We assume:

Condition 1.1. $\{\mathbf{F}_i : i = 1, 2, \dots, m\} \subset \mathcal{S}$ is linearly independent.

If a given SDP does not satisfy this assumption, the SDPA can abnormally stop due to some numerical instability, but it does not mean that it will necessarily happen.

If we deal with a different primal-dual pair \mathcal{P}' and \mathcal{D}' of the form

$$\text{SDP}' \left\{ \begin{array}{l} \mathcal{P}' : \text{minimize } \mathbf{A}_0 \bullet \mathbf{X} \\ \text{subject to } \mathbf{A}_i \bullet \mathbf{X} = b_i \ (i = 1, 2, \dots, m), \ \mathcal{S} \ni \mathbf{X} \succeq \mathbf{O}, \\ \\ \mathcal{D}' : \text{maximize } \sum_{i=1}^m b_i y_i \\ \text{subject to } \sum_{i=1}^m \mathbf{A}_i y_i + \mathbf{Z} = \mathbf{A}_0, \ \mathcal{S} \ni \mathbf{Z} \succeq \mathbf{O}, \end{array} \right.$$

we can easily transform it into the standard form SDP as follows:

$$\begin{array}{ll} -\mathbf{A}_i \ (i = 0, \dots, m) & \longrightarrow \mathbf{F}_i \ (i = 0, \dots, m) \\ -b_i \ (i = 1, \dots, m) & \longrightarrow c_i \ (i = 1, \dots, m) \\ \mathbf{X} & \longrightarrow \mathbf{Y} \\ \mathbf{y} & \longrightarrow \mathbf{x} \\ \mathbf{Z} & \longrightarrow \mathbf{X} \end{array}$$

2.2 Example 1

$$\left. \begin{array}{l}
 \mathcal{P}: \text{ minimize } 48y_1 - 8y_2 + 20y_3 \\
 \text{subject to } \mathbf{X} = \begin{pmatrix} 10 & 4 \\ 4 & 0 \end{pmatrix} y_1 + \begin{pmatrix} 0 & 0 \\ 0 & -8 \end{pmatrix} y_2 + \begin{pmatrix} 0 & -8 \\ -8 & -2 \end{pmatrix} y_3 - \begin{pmatrix} -11 & 0 \\ 0 & 23 \end{pmatrix} \\
 \mathbf{X} \succeq \mathbf{O}. \\
 \mathcal{D}: \text{ maximize } \begin{pmatrix} -11 & 0 \\ 0 & 23 \end{pmatrix} \bullet \mathbf{Y} \\
 \text{subject to } \begin{pmatrix} 10 & 4 \\ 4 & 0 \end{pmatrix} \bullet \mathbf{Y} = 48, \begin{pmatrix} 0 & 0 \\ 0 & -8 \end{pmatrix} \bullet \mathbf{Y} = -8 \\
 \begin{pmatrix} 0 & -8 \\ -8 & -2 \end{pmatrix} \bullet \mathbf{Y} = 20, \mathbf{Y} \succeq \mathbf{O}.
 \end{array} \right\}$$

Here

$$\begin{aligned}
 m &= 3, n = 2, \mathbf{c} = \begin{pmatrix} 48 \\ -8 \\ 20 \end{pmatrix}, \mathbf{F}_0 = \begin{pmatrix} -11 & 0 \\ 0 & 23 \end{pmatrix}, \\
 \mathbf{F}_1 &= \begin{pmatrix} 10 & 4 \\ 4 & 0 \end{pmatrix}, \mathbf{F}_2 = \begin{pmatrix} 0 & 0 \\ 0 & -8 \end{pmatrix}, \mathbf{F}_3 = \begin{pmatrix} 0 & -8 \\ -8 & -2 \end{pmatrix}.
 \end{aligned}$$

The data of this problem is contained in the file “example1.dat” (see Section 4.1).

2.3 Example 2

$$\begin{aligned}
 m &= 5, n = 7, \mathbf{c} = \begin{pmatrix} 1.1 \\ -10 \\ 6.6 \\ 19 \\ 4.1 \end{pmatrix}, \\
 \mathbf{F}_0 &= \begin{pmatrix} -1.4 & -3.2 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -3.2 & -28 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 15 & -12 & 2.1 & 0.0 & 0.0 \\ 0.0 & 0.0 & -12 & 16 & -3.8 & 0.0 & 0.0 \\ 0.0 & 0.0 & 2.1 & -3.8 & 15 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.8 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -4.0 \end{pmatrix}, \\
 \mathbf{F}_1 &= \begin{pmatrix} 0.5 & 5.2 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 5.2 & -5.3 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 7.8 & -2.4 & 6.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -2.4 & 4.2 & 6.5 & 0.0 & 0.0 \\ 0.0 & 0.0 & 6.0 & 6.5 & 2.1 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -4.5 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -3.5 \end{pmatrix}, \\
 &\quad \bullet \\
 &\quad \bullet \\
 &\quad \bullet \\
 \mathbf{F}_5 &= \begin{pmatrix} -6.5 & -5.4 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -5.4 & -6.6 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 6.7 & -7.2 & -3.6 & 0.0 & 0.0 \\ 0.0 & 0.0 & -7.2 & 7.3 & -3.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -3.6 & -3.0 & -1.4 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 6.1 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -1.5 \end{pmatrix}.
 \end{aligned}$$

As shown in this example, the SDPA handles block diagonal matrices. The data of this example is contained in the file “example2.dat” (see Section 4.2).

3. Files Necessary to Execute the SDPA

We need the following files to execute the SDPA:

- “**sdpa**” — The executable binary for solving an SDP.
- “input data file” — Any file name with the postfix “.dat” or “.dat-s” are accepted; for example, “problem.dat” and “example.dat-s” are legitimate names for input files. The SDPA distinguishes a dense input data file with the postfix “.dat” from a sparse input data file with the postfix “.dat-s”. See Sections 4. and 7.2 for details.
- “**param.sdpa**” — The file describing the parameters used in the “sdpa”. See Section 5. for details. The name is fixed to “**param.sdpa**”.
- “output file” — Any file name excepting “**sdpa**” and “**param.sdpa**”. For example, “problem.1” and “example.out” are legitimate names for output files. See Section 6. for more details.

The files “example1.dat” (see Section 4.1) and “example2.dat” (see Section 4.2) contain the input data of Example 1 and Example 2, respectively, which we have stated in the previous section. To solve Example 1, type

```
$ ./sdpa example1.dat example1.out
```

Here “example1.out” denotes an “output file” in which the SDPA stores computational results such as an approximate optimal solution, an approximate optimal value of Example 1, *etc.* Similarly, we can solve Example 2.

4. Input Data File

4.1 “example1.dat” — Input Data File of Example 1

```
"Example 1: mDim = 3, nBLOCK = 1, {2}"
 3 = mDIM
 1 = nBLOCK
 2 = bBLOCKsTRUCT
{48, -8, 20}
{ {-11, 0}, { 0, 23} }
{ { 10, 4}, { 4, 0} }
{ { 0, 0}, { 0, -8} }
{ { 0, -8}, {-8, -2} }
```

4.2 “example2.dat” — Input Data File of Example 2

```
*Example 2:
*mDim = 5, nBLOCK = 3, {2,3,-2}
 5 = mDIM
 3 = nBLOCK
(2, 3, -2) = bBLOCKsTRUCT
{1.1, -10, 6.6, 19, 4.1}
```

```

{
{ { -1.4, -3.2 },
  { -3.2, -28 } }
{ { 15, -12, 2.1 },
  {-12, 16, -3.8 },
  { 2.1, -3.8, 15 } }
{ 1.8, -4.0 }
}
{
{ { 0.5, 5.2 },
  { 5.2, -5.3 } }
{ { 7.8, -2.4, 6.0 },
  {-2.4, 4.2, 6.5 },
  { 6.0, 6.5, 2.1 } }
{ -4.5, -3.5 }
}

```

•
•
•

```

{
{ { -6.5, -5.4 },
  { -5.4, -6.6 } }
{ { 6.7, -7.2, -3.6 },
  {-7.2, 7.3, -3.0 },
  {-3.6, -3.0, -1.4 } }
{ 6.1, -1.5 }
}

```

4.3 Format of the Input Data File

In general, the structure of an input data file is as follows:

Title and Comments

m — number of the primal variables x_i 's

nBLOCK — number of blocks

bBLOCKsSTRUCT — block structure vector

c

F_0

F_1

⋮

F_m

In Sections 4.4 through 4.8, we explain each item of the input data file in details.

4.4 Title and Comments

On the top of the input data file, we can write a single or multiple lines for “Title and Comments”. Each line of “Title and Comments” must begin with " or * and should consist of no more than 75 letters; for example

"Example 1: mDim = 3, nBLOCK = 1, {2}"

in the file “example1.dat”, and

```
*Example 2:
*mDim = 5, nBLOCK = 3, {2,3,-2}
```

in the file “example2.dat”. The SDPA displays “Title and Comments” when it starts. “Title and Comments” can be omitted.

4.5 Number of the Primal Variables

We write the number m of the primal variables in a line following the line(s) of “Title and Comments” in the input data file. All the letters after m through the end of the line are neglected. We have

```
3 = mDIM
```

in the file “example1.dat”, and

```
5 = mDIM
```

in the file “example2.dat”. In either case, the letters “= mDIM” are neglected.

4.6 Number of Blocks and the Block Structure Vector

The SDPA handles block diagonal matrices as we have seen in Section 2.3. We express a common matrix data structure for the constraint matrices $\mathbf{F}_0, \mathbf{F}_1, \dots, \mathbf{F}_m$ using the terms number of blocks, denoted by nBLOCK, and block structure vector, denoted by bBLOCKsTRUCT. If we deal with a block diagonal matrix \mathbf{F} of the form

$$\left. \begin{aligned} \mathbf{F} &= \begin{pmatrix} \mathbf{B}_1 & \mathbf{O} & \mathbf{O} & \dots & \mathbf{O} \\ \mathbf{O} & \mathbf{B}_2 & \mathbf{O} & \dots & \mathbf{O} \\ \vdots & \vdots & \vdots & \ddots & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{O} & \dots & \mathbf{B}_\ell \end{pmatrix}, \\ \mathbf{B}_i &: \text{ a } p_i \times p_i \text{ symmetric matrix } (i = 1, 2, \dots, \ell), \end{aligned} \right\} \quad (1)$$

we define the number nBLOCK of blocks and the block structure vector bBLOCKsTRUCT as follows:

$$\begin{aligned} \text{nBLOCK} &= \ell, \\ \text{bBLOCKsTRUCT} &= (\beta_1, \beta_2, \dots, \beta_\ell), \\ \beta_i &= \begin{cases} p_i & \text{if } \mathbf{B}_i \text{ is a symmetric matrix,} \\ -p_i & \text{if } \mathbf{B}_i \text{ is a diagonal matrix.} \end{cases} \end{aligned}$$

For example, if \mathbf{F} is of the form

$$\begin{pmatrix} 1 & 2 & 3 & 0 & 0 & 0 & 0 \\ 2 & 4 & 5 & 0 & 0 & 0 & 0 \\ 3 & 5 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix}, \quad (2)$$

we have

$$\text{nBLOCK} = 3 \quad \text{and} \quad \text{bBLOCKsTRUCT} = (3, 2, -2)$$

If

$$\mathbf{F} = \begin{pmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{pmatrix}, \text{ where } \star \text{ denotes a real number,}$$

is a usual symmetric matrix with no block diagonal structure, we define

$$\text{nBLOCK} = 1 \text{ and } \text{bBLOCKsTRUCT} = 3$$

We separately write each of nBLOCK and bBLOCKsTRUCT in one line. Any letter after either of nBLOCK and bBLOCKsTRUCT through the end of the line is neglected. In addition to blank letter(s), and the tab code(s), we can use the letters

, () { }

to separate elements of the block structure vector bBLOCKsTRUCT. We have

```
1 = nBLOCK
2 = bBLOCKsTRUCT
```

in Example 1 (see the file “example1.dat” in Section 4.1), and

```
3 = nBLOCK
2 3 -2 = bBLOCKsTRUCT
```

in Example 2 (see the file “example2.dat” in Section 4.2). In either case, the letters “= nBLOCK” and “= bBLOCKsTRUCT” are neglected.

4.7 Constant Vector

Specify all the elements c_1, c_2, \dots, c_m of the cost vector \mathbf{c} . In addition to blank letter(s) and tab code(s), we can use the letters

, () { }

to separate elements of the vector \mathbf{c} . We have

```
{48, -8, 20}
```

in Example 1 (see the file “example1.dat” in Section 4.1), and

```
{1.1, -10, 6.6, 19, 4.1}
```

in Example 2 (see the file “example2.dat” in Section 4.2).

4.8 Constraint Matrices

We describe the constraint matrices $\mathbf{F}_0, \mathbf{F}_1, \dots, \mathbf{F}_m$ according to the data format specified by nBLOCK and bBLOCKsTRUCT stated in Section 4.6. In addition to blank letter(s) and tab code(s), we can use the letters

, () { }

to separate elements of the matrices $\mathbf{F}_0, \mathbf{F}_1, \dots, \mathbf{F}_m$ and their elements. In the general case of the block diagonal matrix \mathbf{F} given in (1), we write the elements of $\mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_\ell$ sequentially; when \mathbf{B}_i is a diagonal matrix, we write only the diagonal elements sequentially. For instance, for the matrix \mathbf{F} given by (2) (nBLOCK = 3, bBLOCKsTRUCT = (3, 2, -2)), the corresponding representation of the matrix \mathbf{F} turns out to be

```
{ {1 2 3} {2 4 5} {3 5 6}}, {1 2} {2 3}, {4, 5} }
```

In Example 1 with nBLOCK = 1 and bLOCKsTRUCT = 2, we have

```
{ {-11, 0}, { 0, 23} }  
{ { 10, 4}, { 4, 0} }  
{ { 0, 0}, { 0, -8} }  
{ { 0, -8}, {-8, -2} }
```

See the file “example1.dat” in Section 4.1.

In Example 2 with nBLOCK = 3 and bLOCKsTRUCT = (2, 3, -2), we have

```
{  
{ { -1.4, -3.2 },  
  { -3.2,-28   } }  
{ { 15, -12, 2.1 },  
  {-12, 16, -3.8 },  
  { 2.1, -3.8, 15   } }  
  { 1.8, -4.0 }  
}  
{  
{ { 0.5, 5.2 },  
  { 5.2, -5.3 } }  
{ { 7.8, -2.4, 6.0 },  
  {-2.4, 4.2, 6.5 },  
  { 6.0, 6.5, 2.1   } }  
  {-4.5, -3.5 }  
}
```

•
•
•

```
{  
{ { -6.5, -5.4 },  
  { -5.4, -6.6 } }  
{ { 6.7, -7.2, -3.6 },  
  {-7.2, 7.3, -3.0 },  
  {-3.6, -3.0, -1.4 } }  
  { 6.1, -1.5 }  
}
```

See the file “example2.dat” in Section 4.2.

Remark. We can also write the input data of Example 1 without using any letters

, () { }

such as

```
"Example 1: mDim = 3, nBLOCK = 1, {2}"
```

```
3  
1  
2  
48 -8 20  
-11 0 0 23  
10 4 4 0  
0 0 0 -8  
0 -8 -8 -2
```

5. Parameter File

First we show the default parameter file “param.sdpa” below.

```

40      unsigned int maxIteration;
1.0E-7  double 0.0 < epsilonStar;
1.0E2   double 0.0 < lambdaStar;
2.0     double 1.0 < omegaStar;
-1.0E5  double lowerBound;
1.0E5   double upperBound;
0.1     double 0.0 <= betaStar < 1.0;
0.2     double 0.0 <= betaBar < 1.0, betaStar <= betaBar;
0.9     double 0.0 < gammaStar < 1.0;
1.0E-7  double 0.0 < epsilonDash;

```

The file “param.sdpa” needs to have these 10 lines which sets 10 parameters. Each line of this file contains one of the 10 parameters followed by an comment. When the SDPA reads the file “param.sdpa”, it neglects the comments.

- maxIteration — Maximum number of iterations. The SDPA stops when the iteration exceeds maxIteration.
- epsilonStar, epsilonDash — The accuracy of an approximate optimal solution of the SDP. When the current iterate $(\mathbf{x}^k, \mathbf{X}^k, \mathbf{Y}^k)$ satisfies all of the inequalities

$$\begin{aligned}
 \text{epsilonDash} &\geq \max \left\{ \left| [\mathbf{X}^k - \sum_{i=1}^m \mathbf{F}_i x_i^k + \mathbf{F}_0]_{pq} \right| : p, q = 1, 2, \dots, n \right\}, \\
 \text{epsilonDash} &\geq \max \left\{ \left| \mathbf{F}_i \bullet \mathbf{Y}^k - c_i \right| : i = 1, 2, \dots, m \right\}, \\
 \text{epsilonStar} &\geq \frac{|\sum_{i=1}^m c_i x_i^k - \mathbf{F}_0 \bullet \mathbf{Y}^k|}{\max \left\{ (|\sum_{i=1}^m c_i x_i^k| + |\mathbf{F}_0 \bullet \mathbf{Y}^k|)/2.0, 1.0 \right\}} \\
 &= \frac{|\text{primal objective value} - \text{dual objective value}|}{\max \{ (|\text{primal objective value}| + |\text{dual objective value}|)/2.0, 1.0 \}},
 \end{aligned}$$

the SDPA stops. Too small epsilonStar and epsilonDash may cause numerical instability. A reasonable choice is epsilonStar and epsilonDash $\geq 1.0E - 7$.

- lambdaStar — This parameter determines an initial point $(\mathbf{x}^0, \mathbf{X}^0, \mathbf{Y}^0)$ such that

$$\mathbf{x}^0 = \mathbf{0}, \quad \mathbf{X}^0 = \text{lambdaStar} \times \mathbf{I}, \quad \mathbf{Y}^0 = \text{lambdaStar} \times \mathbf{I}.$$

Here \mathbf{I} denotes the identity matrix. It is desirable to choose an initial point $(\mathbf{x}^0, \mathbf{X}^0, \mathbf{Y}^0)$ having the same order of magnitude of an optimal solution $(\mathbf{x}^*, \mathbf{X}^*, \mathbf{Y}^*)$ of the SDP. In general, however, choosing such lambdaStar is difficult. If there is no information on the magnitude of an optimal solution $(\mathbf{x}^*, \mathbf{X}^*, \mathbf{Y}^*)$ of the SDP, we strongly recommend to take a sufficiently large lambdaStar such that

$$\mathbf{X}^* \preceq \text{lambdaStar} \times \mathbf{I} \quad \text{and} \quad \mathbf{Y}^* \preceq \text{lambdaStar} \times \mathbf{I}.$$

- omegaStar — This parameter determines the region in which the SDPA searches an optimal solution. For the primal problem \mathcal{P} , the SDPA searches a minimum solution (\mathbf{x}, \mathbf{X}) within the region

$$\mathbf{O} \preceq \mathbf{X} \preceq \text{omegaStar} \times \mathbf{X}^0 = \text{omegaStar} \times \text{lambdaStar} \times \mathbf{I},$$

and stops the iteration if it detects that the primal problem \mathcal{P} has no minimum solution in this region. For the dual problem \mathcal{D} , the SDPA searches a maximum solution \mathbf{Y} within the region

$$\mathbf{O} \preceq \mathbf{Y} \preceq \text{omegaStar} \times \mathbf{Y}^0 = \text{omegaStar} \times \text{lambdaStar} \times \mathbf{I},$$

and stops the iteration if it detects that the dual problem \mathcal{D} has no maximum solution in this region. Again we recommend to take a large lambdaStar and a small omegaStar > 1 .

- lowerBound — Lower bound of the minimum objective value of the primal problem \mathcal{P} . When the SDPA generates a primal feasible solution $(\mathbf{x}^k, \mathbf{X}^k)$ whose objective value $\sum_{i=1}^m c_i x_i^k$ gets smaller than the lowerBound, the SDPA stops the iteration; the primal problem \mathcal{P} is likely to be unbounded and the dual problem \mathcal{D} is likely to be infeasible if the lowerBound is sufficiently small.
- upperBound — Upper bound of the maximum objective value of the dual problem \mathcal{D} . When the SDPA generates a dual feasible solution \mathbf{Y}^k whose objective value $\mathbf{F}_0 \bullet \mathbf{Y}^k$ gets larger than the upperBound, the SDPA stops the iteration; the dual problem \mathcal{D} is likely to be unbounded and the primal problem \mathcal{P} is likely to be infeasible if the upperBound is sufficiently large.
- betaStar — Parameter controlling the search direction when $(\mathbf{x}^k, \mathbf{X}^k, \mathbf{Y}^k)$ is feasible. As we take a smaller betaStar > 0.0 , the search direction tends to get closer to the affine scaling direction without centering.
- betaBar — Parameter controlling the search direction when $(\mathbf{x}^k, \mathbf{X}^k, \mathbf{Y}^k)$ is infeasible. As we take a smaller betaBar > 0.0 , the search direction tends to get closer to the affine scaling direction without centering. The value of betaBar must be no less than the value of betaStar; $0 \leq \text{betaStar} \leq \text{betaBar}$.
- gammaStar — Reduction factor for the primal and dual step lengths; $0.0 < \text{gammaStar} < 1.0$.

6. Output

6.1 Execution of the SDPA

To execute the SDPA, we specify and type the names of three files, “sdpa”, an “input data file” and an “output file” as follows.

```
$ sdpa "input data file" "output file"
```

To solve Example 1, type:

```
$ sdpa example1.dat example1.out
```

6.2 Output on the Display

The SDPA shows some information on the display. In the case of Example 1, we have

```
SDPA start at   Fri Dec  7 18:30:56 2007
data           is example1.dat-s : sparse
parameter is   ./param.sdpa
out            is example.1.out
```

DENSE computations

	mu	thetaP	thetaD	objP	objD	alphaP	alphaD	beta
0	1.0e+04	1.0e+00	1.0e+00	-0.00e+00	+1.20e+03	1.0e+00	9.1e-01	2.00e-01
1	1.6e+03	0.0e+00	9.4e-02	+8.39e+02	+7.51e+01	2.3e+00	9.6e-01	2.00e-01
2	1.7e+02	2.3e-16	3.6e-03	+1.96e+02	-3.74e+01	1.3e+00	1.0e+00	2.00e-01
3	1.8e+01	2.3e-16	1.5e-17	-6.84e+00	-4.19e+01	9.9e-01	9.9e-01	1.00e-01
4	1.9e+00	2.6e-16	1.5e-17	-3.81e+01	-4.19e+01	1.0e-00	1.0e-00	1.00e-01
5	1.9e-01	2.6e-16	3.0e-17	-4.15e+01	-4.19e+01	1.0e-00	9.0e+01	1.00e-01
6	1.9e-02	2.5e-16	1.6e-15	-4.19e+01	-4.19e+01	1.0e-00	1.0e-00	1.00e-01
7	1.9e-03	2.6e-16	2.2e-17	-4.19e+01	-4.19e+01	1.0e-00	1.0e-00	1.00e-01
8	1.9e-04	2.5e-16	1.5e-17	-4.19e+01	-4.19e+01	1.0e-00	1.0e-00	1.00e-01

```

9 1.9e-05 2.7e-16 7.5e-18 -4.19e+01 -4.19e+01 1.0e-00 9.0e+01 1.00e-01
10 1.9e-06 2.6e-16 5.0e-16 -4.19e+01 -4.19e+01 1.0e-00 9.0e+01 1.00e-01

```

```

phase.value = pdOPT
  Iteration = 10
    mu = 1.9180668442024010e-06
relative gap = 9.1554505577840628e-08
  gap = 3.8361336884048019e-06
  digits = 7.0383202783481345e+00
objValPrimal = -4.1899996163866390e+01
objValDual   = -4.189999999999999e+01
p.feas.error = 3.2137639581876834e-14
d.feas.error = 4.7961634663806763e-13
total time   = 0.010
  main loop time = 0.010000
    total time = 0.010000
file  read time = 0.000000
file  read time = 0.000000

```

- mu — The average complementarity $\mathbf{X}^k \bullet \mathbf{Y}^k/n$ (optimality measure). When both \mathcal{P} and \mathcal{D} get feasible, the relation

$$\begin{aligned} \text{mu} &= \left(\sum_{i=1}^m c_i x_i^k - \mathbf{F}_0 \bullet \mathbf{Y}^k \right) / n \\ &= \frac{\text{primal objective function} - \text{dual objective function}}{n} \end{aligned}$$

holds.

- thetaP — The SDPA starts with thetaP = 0.0 if the initial point $(\mathbf{x}^0, \mathbf{X}^0)$ of the primal problem \mathcal{P} is feasible, and thetaP = 1.0 otherwise; hence it usually starts with thetaP = 1.0. In the latter case, the thetaP at the k th iteration is given by

$$\text{thetaP} = \frac{\max \left\{ \left| [\mathbf{X}^k - \sum_{i=1}^m \mathbf{F}_i x_i^k + \mathbf{F}_0]_{p,q} \right| : p, q = 1, 2, \dots, n \right\}}{\max \left\{ \left| [\mathbf{X}^0 - \sum_{i=1}^m \mathbf{F}_i x_i^0 + \mathbf{F}_0]_{p,q} \right| : p, q = 1, 2, \dots, n \right\}};$$

The thetaP is theoretically monotone non-increasing, and when it gets 0.0, we obtain a primal feasible solution $(\mathbf{x}^k, \mathbf{X}^k)$. In the example above, we obtained a primal feasible solution in the 1st iteration.

- thetaD — The SDPA starts with thetaD = 0.0 if the initial point \mathbf{Y}^0 of the dual problem \mathcal{D} is feasible, and thetaD = 1.0 otherwise; hence it usually starts with thetaD = 1.0. In the latter case, the thetaD at the k th iteration is given by

$$\text{thetaD} = \frac{\max \left\{ \left| \mathbf{F}_i \bullet \mathbf{Y}^k - c_i \right| : i = 1, 2, \dots, m \right\}}{\max \left\{ \left| \mathbf{F}_i \bullet \mathbf{Y}^0 - c_i \right| : i = 1, 2, \dots, m \right\}};$$

The thetaD is theoretically monotone non-increasing, and when it gets 0.0, we obtain a dual feasible solution \mathbf{Y}^k . In the example above, we obtained a dual feasible solution in the 3rd iteration.

- objP — The primal objective function value.
- objD — The dual objective function value.
- alphaP — The primal step length.
- alphaD — The dual step length.

- `beta` — The search direction parameter.
- `phase.value` — The status when the iteration stops, taking one of the values `pdOPT`, `noINFO`, `pFEAS`, `dFEAS`, `pdFEAS`, `pdINF`, `pFEAS_dINF`, `pINF_dFEAS`, `pUNBD` and `dUNBD`.

`pdOPT` : The normal termination yielding both primal and dual approximate optimal solutions.

`noINFO` : The iteration has exceeded the `maxIteration` and stopped with no information on the primal feasibility and the dual feasibility.

`pFEAS` : The primal problem \mathcal{P} got feasible but the iteration has exceeded the `maxIteration` and stopped.

`dFEAS` : The dual problem \mathcal{D} got feasible but the iteration has exceeded the `maxIteration` and stopped.

`pdFEAS` : Both primal problem \mathcal{P} and the dual problem \mathcal{D} got feasible, but the iteration has exceeded the `maxIteration` and stopped.

`pdINF` : At least one of the primal problem \mathcal{P} and the dual problem \mathcal{D} is expected to be infeasible. More precisely, there is no optimal solution $(\mathbf{x}, \mathbf{X}, \mathbf{Y})$ of the SDP such that

$$\begin{aligned} \mathbf{O} &\preceq \mathbf{X} \preceq \text{omegaStar} \times \mathbf{X}^0, \\ \mathbf{O} &\preceq \mathbf{Y} \preceq \text{omegaStar} \times \mathbf{Y}^0, \\ \sum_{i=1}^m c_i x_i &= \mathbf{F}_0 \bullet \mathbf{Y}. \end{aligned}$$

`pFEAS_dINF` : The primal problem \mathcal{P} has become feasible but the dual problem is expected to be infeasible. More precisely, there is no dual feasible solution \mathbf{Y} such that

$$\mathbf{O} \preceq \mathbf{Y} \preceq \text{omegaStar} \times \mathbf{Y}^0 = \text{lambdaStar} \times \text{omegaStar} \times \mathbf{I}.$$

`pINF_dFEAS` : The dual problem \mathcal{D} has become feasible but the primal problem is expected to be infeasible. More precisely, there is no feasible solution (\mathbf{x}, \mathbf{X}) such that

$$\mathbf{O} \preceq \mathbf{X} \preceq \text{omegaStar} \times \mathbf{X}^0 = \text{lambdaStar} \times \text{omegaStar} \times \mathbf{I}.$$

`pUNBD` : The primal problem is expected to be unbounded. More precisely, the SDPA has stopped generating a primal feasible solution $(\mathbf{x}^k, \mathbf{X}^k)$ such that

$$\text{objP} = \sum_{i=1}^m c_i x_i^k < \text{lowerBound}.$$

`dUNBD` : The dual problem is expected to be unbounded. More precisely, the SDPA has stopped generating a dual feasible solution \mathbf{Y}^k such that

$$\text{objD} = \mathbf{F}_0 \bullet \mathbf{Y}^k > \text{upperBound}.$$

- `Iteration` — The iteration number when the SDPA terminated.
- `relative gap` — The relative gap

$$\frac{|\text{objP} - \text{objD}|}{\max\{1.0, (|\text{objP}| + |\text{objD}|)/2\}}.$$

This value is compared with `epsilonStar` (Section 5.).

- `gap` — The gap is $\mu \times n$.
- `digits` — This value indicates how `objP` and `objD` resemble by the following definition.

$$\begin{aligned} \text{digits} &= -\log_{10} \frac{|\text{objP} - \text{objD}|}{(|\text{objP}| + |\text{objD}|)/2.0} \\ &= -\log_{10} \frac{|\sum_{i=1}^m c_i x_i^k - \mathbf{F}_0 \bullet \mathbf{Y}^k|}{(|\sum_{i=1}^m c_i x_i^k| + |\mathbf{F}_0 \bullet \mathbf{Y}^k|)/2.0} \end{aligned}$$

- objValPrimal — The primal objective function value

$$\text{objValPrimal} = \sum_{i=1}^m c_i x_i^k.$$

- objValDual — The dual objective function value

$$\text{objValD} = \mathbf{F}_0 \bullet \mathbf{Y}^k.$$

- p.feas.error — This value indicates the primal infeasibility in the last iteration,

$$\text{p.feas.error} = \max \left\{ \left| [\mathbf{X}^k - \sum_{i=1}^m \mathbf{F}_i x_i^k + \mathbf{F}_0]_{p,q} \right| : p, q = 1, 2, \dots, n \right\}$$

This value is compared with **epsilonDash** (Section 5.). Even if the primal problem is feasible, this value may not be 0 due to numerical errors.

- d.feas.error — This value indicates the dual infeasibility in the last iteration,

$$\text{d.feas.error} = \max \left\{ \left| \mathbf{F}_i \bullet \mathbf{Y}^k - c_i \right| : i = 1, 2, \dots, m \right\}.$$

This value is compared with **epsilonDash** (Section 5.). Even if the dual problem is feasible, this value may not be 0 due to numerical errors.

- total time — This value indicates how much time the SDPA needs to execute all subroutines.
- main loop time — This value indicates how much time the SDPA needs between the first iteration and the last iteration.
- file read time — This value is how much time the SDPA needs to read from the input file and store the data in memory.

6.3 Output to a File

We show the content of the file “example2.out” on which the SDPA has written the computational results of Example 2.

```
SDPA start at Fri Dec 7 18:32:10 2007
*Example 2:
*mDim = 5, nBLOCK = 3, {2,3,-2}
data      is example2.dat
parameter is ./param.sdpa
out       is example2.out
  mu      thetaP thetaD objP      objD      alphaP alphaD beta
0 1.0e+04 1.0e+00 1.0e+00 -0.00e+00 +1.44e+03 8.8e-01 6.6e-01 2.00e-01
1 3.3e+03 1.2e-01 3.4e-01 +4.94e+02 +2.84e+02 1.0e+00 8.2e-01 2.00e-01
2 9.0e+02 4.9e-16 6.2e-02 +8.66e+02 -2.60e+00 1.0e+00 1.0e+00 2.00e-01
3 1.4e+02 4.9e-16 8.8e-17 +9.67e+02 +9.12e-01 9.5e-01 5.4e+00 1.00e-01
4 1.8e+01 2.6e-16 8.6e-16 +1.46e+02 +2.36e+01 9.3e-01 1.5e+00 1.00e-01
5 2.7e+00 3.3e-16 4.2e-16 +4.62e+01 +2.74e+01 8.1e-01 1.4e+00 1.00e-01
6 5.7e-01 3.2e-16 2.0e-16 +3.43e+01 +3.03e+01 9.2e-01 9.3e-01 1.00e-01
7 9.5e-02 3.3e-16 4.1e-17 +3.24e+01 +3.18e+01 9.5e-01 9.6e-01 1.00e-01
8 1.3e-02 3.2e-16 1.9e-17 +3.21e+01 +3.20e+01 9.8e-01 1.0e-00 1.00e-01
9 1.5e-03 3.5e-16 2.8e-17 +3.21e+01 +3.21e+01 9.9e-01 1.0e+00 1.00e-01
10 1.5e-04 3.2e-16 1.7e-17 +3.21e+01 +3.21e+01 9.9e-01 1.0e+00 1.00e-01
11 1.5e-05 3.4e-16 3.7e-17 +3.21e+01 +3.21e+01 9.9e-01 1.0e+00 1.00e-01
12 1.5e-06 3.2e-16 4.3e-17 +3.21e+01 +3.21e+01 9.9e-01 1.0e+00 1.00e-01
```

13 1.5e-07 3.3e-16 3.1e-17 +3.21e+01 +3.21e+01 9.9e-01 1.0e+00 1.00e-01

```
phase.value = pdOPT
  Iteration = 13
    mu = 1.5017857203245200e-07
relative gap = 3.2787330975500860e-08
  gap = 1.0512500042271640e-06
  digits = 7.4842939352608049e+00
objValPrimal = 3.2062693405e+01
objValDual   = 3.2062692354e+01
p.feas.error = 3.7747582837e-14
d.feas.error = 5.8841820305e-14
total time   = 0.000
```

Parameters are

```
maxIteration = 100
epsilonStar  = 1.000e-07
lambdaStar   = 1.000e+02
omegaStar    = 2.000e+00
lowerBound   = -1.000e+05
upperBound   = 1.000e+05
betaStar     = 1.000e-01
betaBar      = 2.000e-01
gammaStar    = 9.000e-01
epsilonDash  = 1.000e-07
```

```

                                Time(sec) Ratio(% : MainLoop)
Predictor time = 0.000000, nan
  ... abbreviation ...
Total          = 0.000000, nan
```

```
xVec =
{+1.552e+00,+6.710e-01,+9.815e-01,+1.407e+00,+9.422e-01}
xMat =
{
{ {+6.392e-08,-9.638e-09 },
  {-9.638e-09,+4.539e-08 }  }
{ {+7.119e+00,+5.025e+00,+1.916e+00 },
  {+5.025e+00,+4.415e+00,+2.506e+00 },
  {+1.916e+00,+2.506e+00,+2.048e+00 }  }
{+3.432e-01,+4.391e+00}
}
yMat =
{
{ {+2.640e+00,+5.606e-01 },
  {+5.606e-01,+3.718e+00 }  }
{ {+7.616e-01,-1.514e+00,+1.139e+00 },
  {-1.514e+00,+3.008e+00,-2.264e+00 },
  {+1.139e+00,-2.264e+00,+1.705e+00 }  }
{+4.087e-07,+3.195e-08}
}
  main loop time = 0.000000
    total time = 0.000000
file   read time = 0.000000
```

Now we explain the items that appeared above in the file “example2.out”.

- Lines with start “*” — These lines are comments in “example2.dat”.
- Data, parameter, initial, output — These are the file names we assigned for data, parameter, initial point, and output, respectively.
- Lines between “Predictor time” to “Total time” — These lines display the profile data. These information may help us to tune up the parameters, but the details are rather complicate, because the profile data seriously depends on internal algorithms.
- xVec — Approximate optimal primal variable vector \mathbf{x} .
- xMat — Approximate optimal primal variable matrix \mathbf{X} .
- yMat — Approximate optimal dual variable matrix \mathbf{Y} .

6.4 Printing DIMACS Errors

To display the error measures defined at the 7th DIMACS Implementation Challenge on Semidefinite and Related Optimization Problems [6],

1. Go to the subdirectory where SDPA source code is.
2. Edit the file sdpa_io.cpp, line 22

```
#define DIMACS_PRINT 0
```

```
to
```

```
#define DIMACS_PRINT 1
```

3. Type “make clean”.
4. Type “make” to re-compile SDPA.

The SDPA will output on the display and in the output file the following DIMACS errors at the last iteration:

- Err1 — The relative dual feasibility error on the constraints

$$\frac{\sqrt{\sum_{i=1}^m (\mathbf{F}_i \bullet \mathbf{Y}^k - c_i)^2}}{1 + \max\{|c_i| : i = 1, 2, \dots, m\}}$$

- Err2 — The relative dual feasibility error on the semidefiniteness

$$\max \left\{ 0, \frac{-\lambda_{\min}(\mathbf{Y}^k)}{1 + \max\{|c_i| : i = 1, 2, \dots, m\}} \right\}$$

- Err3 — The relative primal feasibility error on the constraints

$$\frac{\|\mathbf{X}^k - \sum_{i=1}^m \mathbf{F}_i x_i^k + \mathbf{F}_0\|_f}{1 + \max\{|\mathbf{F}_0]_{ij}| : i, j = 1, 2, \dots, n\}}$$

- Err4 — The relative primal feasibility error on the semidefiniteness

$$\max \left\{ 0, \frac{-\lambda_{\min}(\mathbf{X}^k)}{1 + \max\{|\mathbf{F}_0]_{ij}| : i, j = 1, 2, \dots, n\}} \right\}$$

- Err5 — The relative duality gap 1

$$\frac{\sum_{i=1}^m c_i x_i^k - \mathbf{F}_0 \bullet \mathbf{Y}^k}{1 + |\sum_{i=1}^m c_i x_i^k| + |\mathbf{F}_0 \bullet \mathbf{Y}^k|}$$

- Err6 — The relative duality gap 2

$$\frac{\mathbf{X}^k \bullet \mathbf{Y}^k}{1 + |\sum_{i=1}^m c_i x_i^k| + |\mathbf{F}_0 \bullet \mathbf{Y}^k|}$$

where $\|\cdot\|_f$ is a norm defined as a sum of the Frobenius norm of each block diagonal matrix, and $\lambda_{\min}(\cdot)$ is the smallest eigenvalue of a matrix.

7. Advanced Use of the SDPA

7.1 Initial Point

If a feasible interior solution $(\mathbf{x}^0, \mathbf{X}^0, \mathbf{Y}^0)$ is known in advance, we may want to start the SDPA from $(\mathbf{x}^0, \mathbf{X}^0, \mathbf{Y}^0)$. In such a case, we can optionally specify a file which contains the data of a feasible interior solution when we execute the SDPA; for example if we want to solve Example 1 from a feasible interior initial point

$$(\mathbf{x}^0, \mathbf{X}^0, \mathbf{Y}^0) = \left(\left(\begin{array}{c} 0.0 \\ -4.0 \\ 0.0 \end{array} \right), \left(\begin{array}{cc} 11.0 & 0.0 \\ 0.0 & 9.0 \end{array} \right), \left(\begin{array}{cc} 5.9 & -1.375 \\ -1.375 & 1.0 \end{array} \right) \right),$$

type

```
$ sdpa example1.dat example1.out example1.ini
```

Here “example1.ini” denotes an initial point file containing the data of a feasible interior solution:

```
{0.0, -4.0, 0.0}
{ {11.0, 0.0}, {0.0, 9.0} }
{ {5.9, -1.375}, {-1.375, 1.0} }
```

In general, the initial point file can have any name with the postfix “.ini”; for example, “example.ini” is a legitimate initial point file name.

An initial point file contains the data

```
 $\mathbf{x}^0$ 
 $\mathbf{X}^0$ 
 $\mathbf{Y}^0$ 
```

in this order, where the description for the m -dimensional vector \mathbf{x}^0 must follow the same format as the constant vector \mathbf{c} (see Section 4.7), and the description of \mathbf{X}^0 and \mathbf{Y}^0 , the same format as the constraint matrix \mathbf{F}_i (see Section 4.8).

7.2 Sparse Input Data File

In Section 4., we have stated the dense data format for inputting the data $m, n, \mathbf{c} \in \mathbb{R}^m$ and $\mathbf{F}_i \in \mathcal{S}$ ($i = 0, 1, \dots, m$). When not only the constant matrices $\mathbf{F}_i \in \mathcal{S}$ ($i = 0, 1, \dots, m$) are block diagonal, but also each block is sparse, the sparse data format described in this section gives us a compact description of the constant matrices.

A sparse input data file must have a name with the postfix “.dat-s”; for example, “problem.dat-s” and “example.dat-s” are legitimate names for sparse input data files. The SDPA distinguishes a sparse input data file with the postfix “.dat-s” from a dense input data file with the postfix “.dat”.

We show below the file “example1.dat-s”, which contains the data of Example 1 (Section 2.2) in the sparse data format.

```
"Example 1: mDim = 3, nBLOCK = 1, {2}"
  3 = mDIM
  1 = nBLOCK
  2 = bLOCKsTRUCT
48 -8 20
0 1 1 1 -11
0 1 2 2 23
1 1 1 1 10
1 1 1 2 4
2 1 2 2 -8
3 1 1 2 -8
3 1 2 2 -2
```

Compare the dense input data file “example1.dat” described in Section 4.1 with the sparse input data file “example1.dat-s” above. The first 5 lines of the file “example1.dat-s” are the same as those of the file “example1.dat”. Following them, each line of the file “example1.dat-s” describes a single element of a constant matrix \mathbf{F}_i ; the 6th line “0 1 1 1 -11” means that the (1, 1)th element of the 1st block of the matrix \mathbf{F}_0 is -11 , and the 11th line “3 1 1 2 -8” means that the (1, 2)th element of the 1st block of the matrix \mathbf{F}_3 is -8 .

In general, the structure of a sparse input data file is as follows:

Title and Comments

m — the number of the primal variables x_i ’s

nBLOCK — the number of blocks

bBLOCKsTRUCT — the block structure vector

\mathbf{c}

$k_1 \ b_1 \ i_1 \ j_1 \ v_1$

$k_2 \ b_2 \ i_2 \ j_2 \ v_2$

...

$k_p \ b_p \ i_p \ j_p \ v_p$

...

$k_q \ b_q \ i_q \ j_q \ v_q$

Here $k_p \in \{0, 1, \dots, m\}$, $b_p \in \{1, 2, \dots, \text{nBLOCK}\}$, $1 \leq i_p \leq j_p$ and $v_p \in \mathbb{R}$. Each line “ k_p, b_p, i_p, j_p, v_p ” means that the value of the (i_p, j_p) th element of the b_p th block of the constant matrix \mathbf{F}_{k_p} is v_p . If the b_p th block is an $\ell \times \ell$ symmetric (non-diagonal) matrix then (i_p, j_p) must satisfy $1 \leq i_p \leq j_p \leq \ell$; hence only nonzero elements in the upper triangular part of the b_p th block are described in the file. If the b_p th block is an $\ell \times \ell$ diagonal matrix then (i_p, j_p) must satisfy $1 \leq i_p = j_p \leq \ell$.

7.3 Sparse Initial Point File

We show below the file “example1.ini-s”, which contains an initial point data of Example 1 in the sparse data format.

```

0.0 -4.0 0.0
1 1 1 1 11
1 1 2 2 9
2 1 1 1 5.9
2 1 1 2 -1.375
2 1 2 2 1

```

Compare the dense initial point file “example1.ini” described in Section 7.1 with the sparse initial file “example1.ini-s” above. The first line of the file “example1.ini-s” is the same as that of the file “example1.ini”, which describes \mathbf{x}^0 in the dense format. Each line of the rest of the file “example1.ini-s” describes a single element of an initial matrix \mathbf{X}^0 if the first number of the line is 1, or a single element of an initial matrix \mathbf{Y}^0 if the first number of the line is 2; The 2nd line “1 1 1 1 11” means that the (1,1)th element of the 1st block of the matrix \mathbf{X}^0 is 11, the 5th line “2 1 1 2 -1.375” means that the (1,2)th element of the 1st block of the matrix \mathbf{Y}^0 is -1.375 .

A sparse initial point file must have a name with the postfix “.ini-s”; for example, “problem.ini-s” and “example.ini-s” are legitimate names for sparse input data files. The SDPA distinguishes a sparse input data file with the postfix “.ini” from a dense input data file with the postfix “.ini-s”

In general, the structure of a sparse input data file is as follows:

```

 $\mathbf{x}^0$ 
s1 b1 i1 j1 v1
s2 b2 i2 j2 v2
...
sp bp ip jp vp
...
sq bq iq jq vq

```

Here $s_p = 1$ or 2 , $b_p \in \{1, 2, \dots, \text{nBLOCK}\}$, $1 \leq i_p \leq j_p$ and $v_p \in \mathbb{R}$. When $s_p = 1$, each line “ $s_p b_p i_p j_p v_p$ ” means that the value of the (i_p, j_p) th element of the b_p th block of the constant matrix \mathbf{X}^0 is v_p . When $s_p = 2$, the line “ $s_p b_p i_p j_p v_p$ ” means that the value of the (i_p, j_p) th element of the b_p th block of the constant matrix \mathbf{Y}^0 is v_p . If the b_p th block is an $\ell \times \ell$ symmetric (non-diagonal) matrix then (i_p, j_p) must satisfy $1 \leq i_p \leq j_p \leq \ell$; hence only nonzero elements in the upper triangular part of the b_p th block are described in the file. If the b_p th block is an $\ell \times \ell$ diagonal matrix then (i_p, j_p) must satisfy $1 \leq i_p = j_p \leq \ell$.

7.4 Obtaining More Precision on the Approximate Solution

By default, each element of the approximate optimal solution $(\mathbf{x}^k, \mathbf{X}^k, \mathbf{Y}^k)$ saved in the output file (see Section 6.3) has a coefficient with 4 digits in scientific notation.

It is possible to increase its precision.

1. Go to the subdirectory where SDPA source code is.
2. Edit the file `sdpa_struct.cpp`, line 25

```

#define P_FORMAT "%8.3e"

```

to, for instance,

```

#define P_FORMAT "%8.5e"

```

3. Type “make clean”.
4. Type “make” to re-compile SDPA.

Now, each element of the approximate optimal solution will have a coefficient with 6 digits in scientific notation.

7.5 More on Parameter File

We may encounter some numerical difficult during the execution of the SDPA with the default parameter file “param.sdpa”, and/or we may want to solve many easy SDPs with similar data more quickly. In such a case, we need to adjust some of the default parameters: betaStar, betaBar, and gammaStar. We present below two sets of those parameters. The one is the set “Stable_but_Slow” for difficult SDPs, and the other is the set “Unstable_but_Fast” for easy SDPs.

Stable_but_Slow

```
1.0E4 double 0.0 < lambdaStar;
0.10 double 0.0 <= betaStar < 1.0;
0.30 double 0.0 <= betaBar < 1.0, betaStar <= betaBar;
0.80 double 0.0 < gammaStar < 1.0;
```

Unstable_but_Fast

```
0.01 double 0.0 <= betaStar < 1.0;
0.02 double 0.0 <= betaBar < 1.0, betaStar <= betaBar;
0.95 double 0.0 < gammaStar < 1.0;
```

Besides these parameters, the value of the parameter lambdaStar, which determines an initial point $(\mathbf{x}^0, \mathbf{X}^0, \mathbf{Y}^0)$, affects the computational efficiency and the numerical stability. Usually a larger lambdaStar is safe although the SDPA may consume few more iterations.

8. Transformation to the Standard Form of SDP

SDP has many applications, but frequently problems are not described in the standard form of SDP. In this section, we show some examples on how to transform to the standard form of SDP.

8.1 Inequality Constraints

First, we consider the case where inequality constraints are added to the primal standard form of SDP. For example,

$$\left\{ \begin{array}{l} \text{minimize} \quad \sum_{i=1}^m c_i x_i \\ \text{subject to} \quad \mathbf{X} = \sum_{i=1}^m \mathbf{F}_i x_i - \mathbf{F}_0, \quad \mathbf{X} \succeq \mathbf{O}, \\ \quad \quad \quad \sum_{i=1}^m \alpha_i^1 x_i \leq \beta^1, \quad \sum_{i=1}^m \alpha_i^2 x_i \geq \beta^2. \end{array} \right.$$

Here, α_i^1, α_i^2 ($i = 1, \dots, m$), $\beta^1, \beta^2 \in \mathbb{R}$. In this case, we add slack variables (t^1, t^2) .

$$\left\{ \begin{array}{l} \text{minimize} \quad \sum_{i=1}^m c_i x_i \\ \text{subject to} \quad \mathbf{X} = \sum_{i=1}^m \mathbf{F}_i x_i - \mathbf{F}_0, \quad \mathbf{X} \succeq \mathbf{O}, \\ \quad \quad \quad t^1 = \sum_{i=1}^m (-\alpha_i^1) x_i - (-\beta^1), \quad t^2 = \sum_{i=1}^m \alpha_i^2 x_i - \beta^2, \quad (t^1, t^2) \geq 0. \end{array} \right.$$

Hence we can reduce the above problem to the following standard form SDP.

$$\left\{ \begin{array}{l} \text{minimize} \quad \sum_{i=1}^m c_i x_i \\ \text{subject to} \quad \bar{\mathbf{X}} = \sum_{i=1}^m \bar{\mathbf{F}}_i x_i - \bar{\mathbf{F}}_0, \bar{\mathbf{X}} \succeq \mathbf{O}, \end{array} \right.$$

where

$$\bar{\mathbf{F}}_i = \begin{pmatrix} \mathbf{F}_i & & \\ & -\alpha_i^1 & \\ & & \alpha_i^2 \end{pmatrix}, \bar{\mathbf{F}}_0 = \begin{pmatrix} \mathbf{F}_0 & & \\ & -\beta^1 & \\ & & \beta^2 \end{pmatrix}, \bar{\mathbf{X}} = \begin{pmatrix} \mathbf{X} & & \\ & t^1 & \\ & & t^2 \end{pmatrix},$$

nBlock = 2, blockStruct = (n, -2).

Next, we consider the case where inequality constraints are added to the dual standard form of SDP. For example,

$$\left\{ \begin{array}{l} \text{maximize} \quad \mathbf{F}_0 \bullet \mathbf{Y} \\ \text{subject to} \quad \mathbf{F}_1 \bullet \mathbf{Y} = c_1, \mathbf{F}_2 \bullet \mathbf{Y} = c_2, \\ \mathbf{F}_3 \bullet \mathbf{Y} \leq c_3, \mathbf{F}_4 \bullet \mathbf{Y} \leq c_4, \mathbf{F}_5 \bullet \mathbf{Y} \geq c_5, \\ \mathbf{Y} \succeq \mathbf{O}. \end{array} \right.$$

In this case, we add slack variables (t_3, t_4, t_5) to each inequality constraint.

$$\left\{ \begin{array}{l} \text{maximize} \quad \mathbf{F}_0 \bullet \mathbf{Y} \\ \text{subject to} \quad \mathbf{F}_1 \bullet \mathbf{Y} = c_1, \mathbf{F}_2 \bullet \mathbf{Y} = c_2, \\ \mathbf{F}_3 \bullet \mathbf{Y} + t_3 = c_3, \mathbf{F}_4 \bullet \mathbf{Y} + t_4 = c_4, \mathbf{F}_5 \bullet \mathbf{Y} - t_5 = c_5, \\ \mathbf{Y} \succeq \mathbf{O}, \quad (t_3, t_4, t_5) \geq 0. \end{array} \right.$$

Thus we can reduce the above problem to the following standard form SDP.

$$\left\{ \begin{array}{l} \text{maximize} \quad \bar{\mathbf{F}}_0 \bullet \bar{\mathbf{Y}} \\ \text{subject to} \quad \bar{\mathbf{F}}_1 \bullet \bar{\mathbf{Y}} = c_1, \bar{\mathbf{F}}_2 \bullet \bar{\mathbf{Y}} = c_2, \\ \bar{\mathbf{F}}_3 \bullet \bar{\mathbf{Y}} = c_3, \bar{\mathbf{F}}_4 \bullet \bar{\mathbf{Y}} = c_4, \bar{\mathbf{F}}_5 \bullet \bar{\mathbf{Y}} = c_5, \\ \bar{\mathbf{Y}} \succeq \mathbf{O}, \end{array} \right.$$

where

$$\bar{\mathbf{Y}} = \begin{pmatrix} \mathbf{Y} & & & & \\ & t_3 & & & \\ & & t_4 & & \\ & & & t_5 & \\ & & & & t_5 \end{pmatrix}, \bar{\mathbf{F}}_0 = \begin{pmatrix} \mathbf{F}_0 & & & & \\ & 0 & & & \\ & & 0 & & \\ & & & 0 & \\ & & & & 0 \end{pmatrix}, \bar{\mathbf{F}}_1 = \begin{pmatrix} \mathbf{F}_1 & & & & \\ & 0 & & & \\ & & 0 & & \\ & & & 0 & \\ & & & & 0 \end{pmatrix},$$

$$\bar{\mathbf{F}}_2 = \begin{pmatrix} \mathbf{F}_2 & & & & \\ & 0 & & & \\ & & 0 & & \\ & & & 0 & \\ & & & & 0 \end{pmatrix}, \bar{\mathbf{F}}_3 = \begin{pmatrix} \mathbf{F}_3 & & & & \\ & 1 & & & \\ & & 0 & & \\ & & & 0 & \\ & & & & 0 \end{pmatrix}, \bar{\mathbf{F}}_4 = \begin{pmatrix} \mathbf{F}_4 & & & & \\ & 0 & & & \\ & & 1 & & \\ & & & 0 & \\ & & & & 0 \end{pmatrix},$$

$$\bar{\mathbf{F}}_5 = \begin{pmatrix} \mathbf{F}_5 & & & & \\ & 0 & & & \\ & & 0 & & \\ & & & 0 & \\ & & & & -1 \end{pmatrix},$$

nBlock = 2, blockStruct = (n, -3).

8.2 Norm Minimization Problem

Let $\mathbf{G}_i \in \mathbb{R}^{q \times r}$ ($0 \leq i \leq p$). The norm minimization problem is defined as:

$$\begin{aligned} & \text{minimize} && \left\| \mathbf{G}_0 + \sum_{i=1}^p \mathbf{G}_i x_i \right\| \\ & \text{subject to} && x_i \in \mathbb{R} \quad (1 \leq i \leq p). \end{aligned}$$

Here $\|\mathbf{G}\|$ denotes the 2-norm of \mathbf{G} , *i.e.*,

$$\|\mathbf{G}\| = \max_{\|\mathbf{u}\|=1} \|\mathbf{G}\mathbf{u}\| = \text{the square root of the maximum eigenvalue of } \mathbf{G}^T \mathbf{G}.$$

We can reduce this problem to an SDP:

$$\begin{aligned} & \text{minimize} && x_{p+1} \\ & \text{subject to} && \sum_{i=1}^p \begin{pmatrix} \mathbf{O} & \mathbf{G}_i^T \\ \mathbf{G}_i & \mathbf{O} \end{pmatrix} x_i + \begin{pmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{I} \end{pmatrix} x_{p+1} + \begin{pmatrix} \mathbf{O} & \mathbf{G}_0^T \\ \mathbf{G}_0 & \mathbf{O} \end{pmatrix} \succeq \mathbf{O}. \end{aligned}$$

Thus if we take

$$\begin{aligned} m &= p+1, \quad n = r+q, \quad \mathbf{F}_0 = \begin{pmatrix} \mathbf{O} & -\mathbf{G}_0^T \\ -\mathbf{G}_0 & \mathbf{O} \end{pmatrix}, \\ \mathbf{F}_i &= \begin{pmatrix} \mathbf{O} & \mathbf{G}_i^T \\ \mathbf{G}_i & \mathbf{O} \end{pmatrix}, \quad c_i = 0 \quad (1 \leq i \leq p), \\ \mathbf{F}_{p+1} &= \begin{pmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{I} \end{pmatrix}, \quad c_{p+1} = 1, \end{aligned}$$

then we can reformulate the problem as the primal standard form of SDP.

8.3 Linear Matrix Inequality (LMI)

Let $\mathbf{G}_i \in \mathcal{S}^n$ ($0 \leq i \leq p$). We define the linear combinations of the matrices,

$$\mathbf{G}(\mathbf{x}) = \mathbf{G}_0 + \sum_{i=1}^p x_i \mathbf{G}_i,$$

where $\mathbf{x} \in \mathbb{R}^p$. The Linear Matrix Inequality (LMI) [3] is defined as follows

$$\mathbf{G}(\mathbf{x}) \succeq \mathbf{O}.$$

We want to find $\mathbf{x} \in \mathbb{R}^p$ which satisfies the LMI, or detect that any $\mathbf{x} \in \mathbb{R}^p$ cannot satisfy the LMI. Here we introduce an auxiliary variable x_{p+1} and convert the LMI into an SDP.

$$\begin{aligned} & \text{minimize} && x_{p+1} \\ & \text{subject to} && \mathbf{X} = \mathbf{G}_0 + \sum_{i=1}^p x_i \mathbf{G}_i - x_{p+1} \mathbf{I} \succeq \mathbf{O}, \quad \mathbf{x} \in \mathbb{R}^p \end{aligned}$$

We can reduce this LMI to the primal standard form of SDP if we take

$$m = p+1, \quad \mathbf{F}_0 = -\mathbf{G}_0, \quad \mathbf{F}_i = \mathbf{G}_i, \quad c_i = 0 \quad (1 \leq i \leq p), \quad \mathbf{F}_{p+1} = -\mathbf{I}, \quad c_{p+1} = 1.$$

One important point of this SDP is that it always has a feasible solution. When we can get the optimal solution with $x_{p+1} \geq 0$, then (x_1, \dots, x_p) satisfies the LMI. On the other hand, if $x_{p+1} < 0$ then we can conclude that the LMI cannot be satisfied by any $\mathbf{x} \in \mathbb{R}^p$.

8.4 SDP Relaxation of the Maximum Cut Problem

Let $G = (V, E)$ be a complete undirected graph with a vertex set $V = \{1, 2, \dots, n\}$ and an edge set $E = \{(i, j) : i, j \in V, i < j\}$. We assign a weight $W_{ij} = W_{ji}$ to each edge $(i, j) \in E$. The maximum cut problem is to find a partition (L, R) of V that maximizes the *cut* $w(L, R) = \sum_{i \in L, j \in R} W_{ij}$. Introducing a variable vector $\mathbf{u} \in \mathbb{R}^n$, we can formulate the problem as a nonconvex quadratic program:

$$\text{maximize} \quad \frac{1}{4} \sum_{i=1}^n \sum_{j=1}^n W_{ij} (1 - u_i u_j) \quad \text{subject to} \quad u_i^2 = 1 \quad (1 \leq i \leq n).$$

Here each feasible solution $\mathbf{u} \in \mathbb{R}^n$ of this problem corresponds to a cut (L, R) with $L = \{i \in V : u_i = -1\}$ and $R = \{i \in V : u_i = 1\}$. If we define \mathbf{W} to be the $n \times n$ symmetric matrix with elements $W_{ji} = W_{ij}$ ($(i, j) \in E$) and $W_{ii} = 0$ ($1 \leq i \leq n$), and the $n \times n$ symmetric matrix $\mathbf{C} \in \mathcal{S}^n$ by $\mathbf{C} = \frac{1}{4}(\text{diag}(\mathbf{W}\mathbf{e}) - \mathbf{W})$, where $\mathbf{e} \in \mathbb{R}^n$ denotes the vector of ones and $\text{diag}(\mathbf{W}\mathbf{e})$ the diagonal matrix of the vector $\mathbf{W}\mathbf{e} \in \mathbb{R}^n$, we can rewrite the quadratic program above as

$$\text{maximize} \quad \mathbf{x}^T \mathbf{C} \mathbf{x} \quad \text{subject to} \quad x_i^2 = 1 \quad (1 \leq i \leq n).$$

If $\mathbf{x} \in \mathbb{R}^n$ is a feasible solution of the latter quadratic program, then the $n \times n$ symmetric and positive semidefinite matrix \mathbf{X} whose (i, j) th element X_{ij} is given by $X_{ij} = x_i x_j$ satisfies $\mathbf{C} \bullet \mathbf{X} = \mathbf{x}^T \mathbf{C} \mathbf{x}$ and $X_{ii} = 1$ ($1 \leq i \leq n$). This leads to the following semidefinite programming relaxation of the maximum cut problem:

$$\begin{aligned} & \text{maximize} && \mathbf{C} \bullet \mathbf{X} \\ & \text{subject to} && \mathbf{E}_{ii} \bullet \mathbf{X} = 1 \quad (1 \leq i \leq n), \quad \mathbf{X} \succeq \mathbf{O}. \end{aligned} \tag{3}$$

Here \mathbf{E}_{ii} denotes the $n \times n$ symmetric matrix with (i, i) th element 1 and all others 0.

8.5 Choosing Between the Primal and Dual Standard Forms

Any SDP can be formulated as a primal standard form \mathcal{P} or as a dual standard form \mathcal{D} (see Section 2.1). This does not mean that one is the dual of the other, but simply that they are indeed two different formulations of the same problem, each one having its dual counterpart.

Many times, it is advantageous to choose between the primal and the dual standard forms since one of the formulations is more natural, has a smaller size, it is faster to solve, or it avoids numerical instability on the software.

Let us consider as an example the SDP relaxation of the maximum cut problem (Section 8.4). We formulated this SDP as a dual standard form \mathcal{D} where the problem size is:

$$\begin{aligned} m &= n \\ \text{nBLOCK} &= 1 \\ \text{bBLOCKsTRUCT} &= n. \end{aligned}$$

We can also formulate (mathematically) the same problem as a primal standard form \mathcal{P} , too.

Let \mathbf{H}_{ij} an $n \times n$ symmetric matrix with (i, j) th and (j, i) th element(s) 1 and all others 0. Problem (3) is equivalent to

$$\begin{aligned} & \text{minimize} && -2 \sum_{i=1}^n \sum_{j>i}^n C_{ij} x_{ij} - \sum_{i=1}^n C_{ii} x_{ii} \\ & \text{subject to} && \sum_{i=1}^n \sum_{j=i}^n \mathbf{H}_{ij} x_{ij} \succeq \mathbf{O}, \\ & && x_{ii} \geq 1 \quad (1 \leq i \leq n), \\ & && x_{ii} \leq 1 \quad (1 \leq i \leq n), \end{aligned} \tag{4}$$

where x_{ij} ($1 \leq i \leq n$, $i \leq j \leq n$) is a variable vector now.

This SDP is in primal standard form \mathcal{P} and its size is

$$\begin{aligned} m &= \frac{n(n+1)}{2} \\ \text{nBLOCK} &= 2 \\ \text{bBLOCKsTRUCT} &= (n, -2n) \end{aligned}$$

which seems less advantageous than the dual standard from \mathcal{D} (3). Also the problem (4) in primal standard form \mathcal{P} does not have a strict feasible solution which many cause numerical instability.

In the case of $n = 100$, a typical running time for the formulation (3) in dual standard form is 0.18s, while for the for the formulation (4) in primal standard form is 201.3s.

In some cases, however, a slight increase in the size of the problem can be advantageous if the new formulation has more sparsity in its data.

9. For the SDPA 6.2.1 Users

The major differences between the SDPA 6.2.1 and the SDPA 7.0.5 are the followings:

- The source code was completely revised, unnecessary variables were eliminated, and auxiliary variables re-used. Consequently, the memory usage became less than half of the previous version.
- It utilizes the sparse Cholesky factorization when the Schur Complement Matrix becomes sparse. For that, it uses the SPOOLES library for sparse matrices [2] to obtain an ordering of rows/columns which possibly produces lesser fill-in. Now, the SDPA can solve much more efficiently SDPs with
 - multiple block diagonal matrices
 - multiple non-negative constraints
- There is a modification in the control subroutine in the interior-point algorithm which improved its numerical stability when compared to the previous version.

Though it should be noted there is a remaining problem:

- The current version does not have a callable library interface as SDPA 6.2.1 does, but it will be implemented in the updated version of the SDPA.

References

- [1] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, “LAPACK Users’ Guide, Third Edition” *Society for Industrial and Applied Mathematics (1999)* Philadelphia, PA.
- [2] C. Ashcraft, D. Pierce, D. K. Wah, and J. Wu, “The reference manual for SPOOLES, release 2.2: An object oriented software library for solving sparse linear systems of equations,” Boeing Shared Service Group. Seattle, WA, January 1999. Available at <http://www.netlib.org/linalg/spooles/spooles.2.2.html>.
- [3] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, “Linear Matrix Inequalities in System and Control Theory” *Society for Industrial and Applied Mathematics (1994)* Philadelphia, PA.

- [4] K. Fujisawa, M. Kojima, and K. Nakata, “Exploiting sparsity in primal-dual interior-point methods for semidefinite programming,” *Mathematical Programming, Series B* **79** (1997) 235–253.
- [5] K. Fujisawa, K. Nakata, M. Yamashita, and M. Fukuda, “SDPA project: Solving large-scale semidefinite programs,” *Journal of Operations Research Society of Japan* **50** (2007) 278–298.
- [6] H. D. Mittelmann, “An independent benchmarking of SDP and SOCP solvers,” *Mathematical Programming, Series B* **95** (2003) 407–430.