Research Reports on Mathematical and Computing Sciences
Series B : Operations Research Department of Mathematical and Computing Sciences

Tokyo Institute of Technology
2-12-1-W8-29 Oh-Okayama, Meguro-ku, Tokyo 152-8552, Japan

SDPA-M (SemiDefinite Programming Algorithm in MATLAB) User's Manual — Version 6.2.0

K. Fujisawa[⋆], Y. Futakata[♭], M. Kojima[†], S. Matsuyama,
S. Nakamura, K. Nakata[‡] and M. Yamashita[♯]

B-359, January 2000
Revised: May 2005

Abstract. The SDPA-M (Semidefinite Programming Algorithm in MATLAB) Version 6.2.0 is a MATLAB interface of the SDPA Version 6.2.0 [1], which is known as a fast and numerically stable solver for SDPs (semidefinite programs) [3, 4]. The SDPA-M inherits various features from the SDPA. Particularly, the SDPA-M can read SDPA dense and sparse format input data files of SDPs. In addition, users can easily manipulate and transform their own problems in the MATLAB language, and then solve them by the SDPA-M. This manual and the SDPA-M can be found at the WWW site:

http://grid.r.dendai.ac.jp/sdpa/

Key words Semidefinite Programming, Interior-Point Method, Computer Software, MATLAB

⋆ e-mail:fujisawa@r.dendai.ac.jp

♭ e-mail:YoshiakiFutakata@virginia.edu

† e-mail:kojima@is.titech.ac.jp

‡ e-mail:knakata@me.titech.ac.jp

♯ e-mail:Makoto.Yamashita@ie.kanagawa-u.ac.jp

Contents

1. What's New?

SDPA-M version 6.2.0 provides the MATLAB interface for SDPA version 6.2.0. The big change from the previous version is that users don't have to install ATLAS and CLAPACK anymore; users can use the built-in LAPACK and BLAS in MATLAB. We still provide the version which uses the self-compiled ATLAS and CLAPACK as the previous versions. But we only support the Linux system for the self-compiled ATLAS and CLAPACK version.

Another change from the previous version is that "`param.m`" is written as M-function instead of M-script. The change of the interface is only in this part and we checked that YALMIP[2] worked correctly in its test program. (YALMIP provides an advanced MATLAB interface for using several numerical optimization programs and SDPA-M is one of the solvers which YALMIP supports.)

Finally, we added "`gensdpafile.m`" for reducing our support difficulties. This new function creates an SDP problem file with SDPA sparse format. For example, after users create an SDP problem in MATLAB with

mDIM, nBLOCK, bLOCKsTRUCT, c, F

variables, users create an SDP problem file with SDPA sparse format by

```
>> gensdpafile('sdp.dat-s',mDIM,nBLOCK,bLOCKsTRUCT,c,F)
```

Users can find the file "`sdp.dat-s`" in the current directory. Sending this file to us will make our support easier.

2. Installation

The SDPA-M package is available at the following WWW site:

http://grid.r.dendai.ac.jp/sdpa/

You can select both the binary and source packages. If you are not good at compiling source files, please download the binary package. In the following explanation, we write the command prompt as '`$`'.

2.1. Binary Package

▶ **Windows**:
Download the following package from the above site:

- sdpam.6.2.0.bin.zip

After unzipping the package, copy the following 6 files into the folder in which you want to install SDPA-M:

gensdpafile.m, initial_point.m, param.m, read_data.m, sdpam.m, mexsdpa.dll

After copying these files, read Section.2.4. to finish the installation completely.

▶ **Linux, Solaris or Mac OS X**:
Download the following package from the above site:

- sdpam.6.2.0.bin.tar.gz

To extract the package, if your system is Linux or Mac OS X, enter

```
$ tar zxvf sdpam.6.2.0.bin.tar.gz
```

and if your system is Solaris, enter

```
$ gzip -cd sdpam.6.2.0.bin.tar.gz | tar -xvf -
```

To install SDPA-M for your system, enter

```
$ cd sdpam; /bin/sh install.sh <INSTALL_DIR>
```

Replace `<INSTALL_DIR>` with the directory name in which you want to install SDPA-M. For example, if you want to install SDPA-M in /home/user1/matlab/sdpam, enter

```
$ cd sdpam; /bin/sh install.sh /home/user1/matlab/sdpam
```

When you finish the above operations, read Section.2.4. to finish the installation completely.

2.2. Source Package

▶ **Windows**:
Download the following package from the above site:

- sdpam.6.2.0.src.zip

Note that to compile this package you need Microsoft C++ compiler; therefore if you don't have the compiler, please install a binary package.

After unzipping the package, run a command prompt application and move to "`mex`" folder in "`sdpam`" which is the top folder of this package. If you are in the folder "`sdpam`", enter

```
$ cd mex
```

Before compiling the package, please see the line 11 in `Make.win32`. It is a folder name where you installed MATLAB.

```
Line 11:  MATLAB_ROOT=C:\MATLAB7
```

If you installed MATLAB in `C:\Program Files\MATLAB7`, modify it as

```
Line 11:  MATLAB_ROOT=C:\Program Files\MATLAB7
```

After checking that the line 11 in `Make.win32` is correct, to compile source files, enter

```
$ nmake -f Make.win32
```

If you finish the compile successfully, you can find "`mexsdpam.dll`" in the same folder. Next step is to copy the following 6 files into the folder in which you want to install SDPA-M:

gensdpafile.m, initial_point.m, param.m, read_data.m, sdpam.m, mexsdpa.dll

Note that M-files are in the top folder of the package, but MEX-file is in the "`mex`" folder. After copying these files, read Section.2.4. to finish the installation completely.

► **Linux**:
Download the following package from the above site:

- sdpam.6.2.0.src.tar.gz

To extract the package, enter

```
$ tar zxvf sdpam.6.2.0.src.tar.gz
```

After extracting the package, you have to modify 2 Makefiles. First, in the top directory, you can find "`Makefile`" and there is a line:

```
Line 12:  INSTALL_DIR=$(HOME)/work/sdpam
```

In this line, you can select the directory in which you want to install SDPA-M. Therefore, if you want to install it in "`/home/user1/matlab/sdpam`", replace this line with

```
Line 12:  INSTALL_DIR=/home/user1/matlab/sdpam
```

Next, in "`mex`" directory you can find "`Make.linux`". Copy this file to "`Makefile`", that is

```
$ cp Make.linux Makefile
```

The last step is to compile and install these files. After moving the top directory, enter

```
$ make; make install
```

If you finish the above operations successfully, read Section.2.4. to finish the installation completely. If you have a problem in compiling source files, please check that the command path for "`mex`" command exists. To check it out, enter

```
$ which mex
```

If there is no path, modify "`Makefile`" in "`mex`" directory

```
Line 7:  MEX=mex
```

For example, if your `mex` command exists in "`/opt/matlab/bin`", modify it as

```
MEX=/opt/matlab/bin/mex
```

and repeat

```
$ make; make install
```

► **Solaris**:
Download the following package from the above site:

- sdpam.6.2.0.src.tar.gz

To extract the package, enter

```
$ gzip -cd sdpam.6.2.0.src.tar.gz | tar -xvf -
```

To compile source files, you need SUN C++ compiler. So, if you don't have the compiler, please use a binary package.

After extracting the package, you have to modify 2 Makefiles. First, in the top directory, you can find "`Makefile`" and there is a line:

```
Line 12:  INSTALL_DIR=$(HOME)/work/sdpam
```

In this line, you can select the directory in which you want to install SDPA-M. Therefore, if you want to install it in "`/home/user1/matlab/sdpam`", replace this line with

```
INSTALL_DIR=/home/user1/matlab/sdpam
```

Next, in "`mex`" directory you can find "`Make.solaris`". Copy this file to "`Makefile`", that is

```
$ cp Make.solaris Makefile
```

The last step is to compile and install these files. After moving the top directory, enter

```
$ make; make install.solaris
```

If you finish the above operations successfully, read Section.2.4. to complete the installation. If you have a problem in compiling source files, please check your "`mex`" command path as in the Linux installation guide.

▶ **Mac OS X**:
Download the following package from the above site:

- sdpam.6.2.0.src.tar.gz

To extract the package, enter

```
$ tar zxvf sdpam.6.2.0.src.tar.gz
```

After extracting the package, you have to modify 2 Makefiles. First, in the top directory, you can find "`Makefile`" and there is a line:

```
Line 12:  INSTALL_DIR=$(HOME)/work/sdpam
```

In this line, you can select the directory in which you want to install SDPA-M. Therefore, if you want to install it in "`/home/user1/matlab/sdpam`", replace this line with

```
INSTALL_DIR=/home/user1/matlab/sdpam
```

Next, in "`mex`" directory you can find "`Make.macX`". Copy this file to "`Makefile`", that is

```
$ cp Make.macX Makefile
```

The last step is to compile and install these files. After moving the top directory, enter

```
$ make; make install
```

If you finish the above operations successfully, read Section.2.4. to complete the installation. If you have a problem in compiling source files, please check your "`mex`" command path as in the Linux installation guide.

2.3. Using Self-compiled ATLAS and CLAPACK

For installing the self-compiled ATLAS and CLAPACK version, you have to install SDPA version 6.2.0 beforehand, and please check that the installation of SDPA was correctly done. For installing this package, download the following package from the above site:

- sdpam.6.2.0.src.tar.gz

To extract the package, enter

```
$tar zxvf sdpam.6.2.0.src.tar.gz
```

To install the package, you have to change 2 Makefiles. At the top directory of the package , you can find "`Makefile`"and there is a line:

```
Line 12:  INSTALL_DIR=$(HOME)/work/sdpam
```

In this line, you can select the directory in which you want to install SDPA-M. Therefore, if you want to install it in "`/home/user1/matlab/sdpam`", replace this line with

```
INSTALL_DIR=/home/user1/matlab/sdpam
```

Next, in "`mex`" directory you can find "`Make.extatlas`". Copy this file to "`Makefile`", that is

```
$ cp Make.extatlas Makefile
```

There are 2 lines to modify in this file:

```
Line 7:  LAPACK=$(HOME)/lapack
Line 8:  SDPA=$(HOME)/sdpa
```

Each line implies where ATLAS, CLAPACK and SDPA exist. For example, if you installed ATLAS and CLAPACK in

```
/home/user1/lapack
```

and SDPA in

```
/home/user1/sdpa
```

Replace these lines with

```
LAPACK=/home/user1/lapack
SDPA=/home/user1/sdpa
```

The last step is to compile and install these files. After moving the top directory, enter

```
$ make; make install
```

If you finish the above operations successfully, read Section.2.4. to complete the installation. If you have a problem in compiling source files, please check your "`mex`" command path as in the Linux installation guide.

2.4. After Installation

After installing the necessary files, we recommend you to add the path for SDPA-M to the MATLAB search path by adding the following line:

```
addpath <INSTALL_DIR>
```

to

```
$MATLAB/toolbox/local/startup.m
```

for Windows, and to

```
$HOME/matlab/startup.m
```

for Linux, Solaris, or Mac OS X. Here "`<INSTALL_DIR>`" means the path in which you installed SDPA-M and `$MATLAB` is the folder in which MATLAB is installed. For examples, if you installed SDPA-M in

```
/home/user1/matlab/sdpam
```

add the line into "`$HOME/matlab/startup.m`":

```
addpath /home/user1/matlab/sdpam
```

Now you have completed the installation of SDPA-M. Check whether the installation is finished successfully by solving a simple SDP (See Section.4.).

3. Semidefinite Program

3.1. Standard Form SDP and Its Dual

The SDPA-M solves the following standard form semidefinite program [3, 4] and its dual. Here

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

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

$\boldsymbol{F}_i \in \mathcal{S}$ $(i = 0, 1, 2, \ldots, m)$: constraint matrices.

$\boldsymbol{O} \in \mathcal{S}$: the zero matrix.

$\boldsymbol{c} = \begin{pmatrix} c_1 \\ c_2 \\ \cdot \\ c_m \end{pmatrix} \in R^m$: a cost vector, $\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ x_m \end{pmatrix} \in R^m$: a variable vector,

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

$\boldsymbol{U} \bullet \boldsymbol{V}$: the inner product of $\boldsymbol{U}$, $\boldsymbol{V} \in \mathcal{S}$, i.e., $\displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} U_{ij} V_{ij}$

$\boldsymbol{U} \succeq \boldsymbol{O}$, $\iff$ $\boldsymbol{U} \in \mathcal{S}$ is positive semidefinite.

Throughout this manual, we denote the primal-dual pair of $\mathcal{P}$ and $\mathcal{D}$ by the SDP. The SDP is determined by m, n, $\boldsymbol{c} \in R^m$, $\boldsymbol{F}_i \in \mathcal{S}$ $(i = 0, 1, 2, \ldots, m)$. When $(\boldsymbol{x}, \boldsymbol{X})$ is a feasible solution (or a minimum solution, respectively) of the primal problem $\mathcal{P}$ and $\boldsymbol{Y}$ is a feasible solution (or a maximum solution, respectively) of the dual problem $\mathcal{D}$, we call $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y})$ a feasible solution (or an optimal solution, respectively) of the SDP.

We assume:

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

If the SDP did not satisfy this assumption, it might cause some trouble (numerical instability) that would abnormally stop the execution of the SDPA-M.

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

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

we can easily transform the SDP' into the SDP as follows:

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

3.2. Example 1

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

Here

$$m \ = \ 3, \ n = 2, \ \boldsymbol{c} = \begin{pmatrix} 48 \\ -8 \\ 20 \end{pmatrix}, \ \boldsymbol{F}_0 = \begin{pmatrix} -11 & 0 \\ 0 & 23 \end{pmatrix},$$

$$F_1 = \begin{pmatrix} 10 & 4 \\ 4 & 0 \end{pmatrix}, \; F_2 = \begin{pmatrix} 0 & 0 \\ 0 & -8 \end{pmatrix}, \; F_3 = \begin{pmatrix} 0 & -8 \\ -8 & -2 \end{pmatrix}.$$

The data (see Section 5.3.) of this problem is contained in the file "example1.dat".

3.3. Example 2

$$m = 5, \; n = 7, \; \boldsymbol{c} = \begin{pmatrix} 1.1 \\ -10 \\ 6.6 \\ 19 \\ 4.1 \end{pmatrix},$$

$$\boldsymbol{F}_0 = \left(\begin{array}{rr|rrr|rr} -1.4 & -3.2 & 0 & 0 & 0 & 0 & 0 \\ -3.2 & -28 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 15 & -12 & 2.1 & 0 & 0 \\ 0 & 0 & -12 & 16 & -3.8 & 0 & 0 \\ 0 & 0 & 2.1 & -3.8 & 15 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -4.0 \end{array}\right),$$

$$\boldsymbol{F}_1 = \left(\begin{array}{rr|rrr|rr} 0.5 & 5.2 & 0 & 0 & 0 & 0 & 0 \\ 5.2 & -5.3 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 7.8 & -2.4 & 6.0 & 0 & 0 \\ 0 & 0 & -2.4 & 4.2 & 6.5 & 0 & 0 \\ 0 & 0 & 6.0 & 6.5 & 2.1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & -4.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -3.5 \end{array}\right)$$

$$\vdots$$

$$\boldsymbol{F}_5 = \left(\begin{array}{rr|rrr|rr} -6.5 & -5.4 & 0 & 0 & 0 & 0 & 0 \\ -5.4 & -6.6 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 6.7 & -7.2 & -3.6 & 0 & 0 \\ 0 & 0 & -7.2 & 7.3 & -3.0 & 0 & 0 \\ 0 & 0 & -3.6 & -3.0 & -1.4 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 6.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1.5 \end{array}\right).$$

As shown in this example, the SDPA-M handles block diagonal matrices. The data (see Section 5.3.) of this example is contained in the file "example2.dat".

4. Quick Start

At first, let's take a simple example to confirm that SDPA-M is correctly installed and to know the basic usage of SDPA-M.

```
>> % read a problem data from the data file 'example1.dat'.
>> [mDIM,nBLOCK,bLOCKsTRUCT,c,F] = read_data('example1.dat');
>>
>> % read an initial point from the initial point file 'example1.ini'.
>> [x0,X0,Y0] = initial_point('example1.ini',mDIM,nBLOCK,bLOCKsTRUCT);
>>
>> % next command is to redirect the output into the file named 'example1.out'.
>> % default output is on display.
>> OPTION = param('print','example1.out');
>>
>> % solve the problem using sdpam.
>> [objVal,x,X,Y,INFO] = sdpam(mDIM,nBLOCK,bLOCKsTRUCT,c,F,x0,X0,Y0,OPTION);
```

Now we have output arguments `objVal,x,X,Y`, `INFO`, and generated an output file named "example1.out". To see the file "example1.out"

```
>> type example1.out

SDPA library start ... (built at May 23 2005 14:54:57)

   mu      thetaP  thetaD  objP      objD      alphaP  alphaD  beta
 0 1.0e+04 1.0e+00 1.0e+00 +3.20e+01 -4.19e+01 1.0e+00 9.1e-01 2.00e-01
 1 1.6e+03 1.0e-16 9.4e-02 +8.39e+02 +7.51e+01 2.3e+00 9.6e-01 2.00e-01
 2 1.7e+02 1.9e-16 3.6e-03 +1.96e+02 -3.74e+01 1.3e+00 1.0e+00 2.00e-01
 3 1.8e+01 1.6e-16 2.2e-17 -6.84e+00 -4.19e+01 9.9e-01 9.9e-01 1.00e-01
 4 1.9e+00 1.6e-16 1.5e-17 -3.81e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01
 5 1.9e-01 1.6e-16 1.5e-17 -4.15e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01
 6 1.9e-02 1.6e-16 7.5e-18 -4.19e+01 -4.19e+01 1.0e-00 9.0e+01 1.00e-01
 7 1.9e-03 1.6e-16 5.8e-16 -4.19e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01
 8 1.9e-04 1.8e-16 2.2e-17 -4.19e+01 -4.19e+01 1.0e-00 9.0e+01 1.00e-01
 9 1.9e-05 1.5e-16 3.1e-15 -4.19e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01
10 1.9e-06 1.6e-16 2.2e-17 -4.19e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01

phase.value = pdOPT
   Iteration = 10
          mu = 1.9180668442025276e-06
relative gap = 9.1554506595324411e-08
         gap = 3.8361336884050552e-06
      digits = 7.0383202735216370e+00
objValPrimal = -4.1899996163866327e+01
objValDual   = -4.1899999999999977e+01
p.feas.error = 1.9317880628477724e-14
d.feas.error = 2.1316282072803006e-14
```

In the following sections, each arguments and contents of the output are explained.

5. Problem Data Input

5.1. Input Arguments

- mDIM — The number of primal variables. We have

  ```
  >> mDIM = 3;
  ```

  in Example1, and

  ```
  >> mDIM = 5;
  ```

  in Example2.

- nBLOCK, bLOCKsTRUCT — The number of blocks, and the block structure vector. The SDPA-M handles block diagonal matrices as we have seen in Section 4. In terms of the number of blocks denoted by nBLOCK, and the block structure vector denoted by bLOCK-sTRUCT, we express a common matrix data structure for the constraint matrices $\boldsymbol{F}_0, \boldsymbol{F}_1, \ldots, \boldsymbol{F}_m$. If we deal with a block diagonal matrix $\boldsymbol{F}$ of the form

  $$\left.\begin{array}{rcl} \boldsymbol{F} & = & \begin{pmatrix} \boldsymbol{B}_1 & \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{B}_2 & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \cdot & \cdot & \cdot & \cdots & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{B}_\ell \end{pmatrix}, \\ \boldsymbol{B}_i & : & \text{a } p_i \times p_i \text{ symmetric matrix} \ \ (i = 1, 2, \ldots, \ell), \end{array}\right\} \tag{1}$$

  we define the number nBLOCK and the block structure vector bLOCKsTRCTURE as follows:

  $$\begin{array}{rcl} \text{nBLOCK} & = & \ell, \\ \text{bLOCKsTRUCT} & = & (\beta_1,\ \beta_2,\ \ldots,\ \beta_\ell), \\ \beta_i & = & \begin{cases} p_i & \text{if } \boldsymbol{B}_i \text{ is a } p_i \times p_i \text{ symmetric matrix,} \\ -p_i & \text{if } \boldsymbol{B}_i \text{ is a } p_i \times p_i \text{ diagonal matrix.} \end{cases} \end{array}$$

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

  $$\left(\begin{array}{ccc|cc|cc} 1 & 2 & 3 & 0 & 0 & 0 & 0 \\ 2 & 4 & 5 & 0 & 0 & 0 & 0 \\ 3 & 5 & 6 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 3 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{array}\right), \tag{2}$$

  we have

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

  If

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

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

$$\text{nBLOCK} = 1 \quad \text{and} \quad \text{bLOCKsTRUCT} = 3$$

We have

```
>> nBLOCK = 1;
>> bLOCKsTRUCT = 2;
```

in Example 1, and

```
>> nBLOCK = 3;
>> bLOCKsTRUCT = [ 2 3 -2 ];
```

in Example 2. bLOCKsTRUCT can be a column vector.

- $\boldsymbol{c}$ — Constant vector. We write all the elements $c_1, c_2, \ldots, c_m$ of the constant vector $\boldsymbol{c}$. We have

```
>> c = [ 48 -8 20 ];
```

in Example 1, and

```
>> c = [ 1.1 -10 6.6 19 4.1 ];
```

in Example 2.

- $\boldsymbol{F}$ — Constraint matrices. $\boldsymbol{F}$ is nBLOCK×(mDIM+1) cell matrix.

	$\boldsymbol{F}_0$	$\boldsymbol{F}_1$			$\boldsymbol{F}_m$
1st block	F{1,1}	F{1,2}	⋯	⋯	F{1,m+1}
2nd block	F{2,1}	F{2,2}	⋯	⋯	F{2,m+1}
	⋮	⋮	⋮	⋮	⋮
	⋮	⋮	⋮	⋮	⋮
n-th block	F{n,1}	F{n,2}	⋯	⋯	F{n,m+1}

Here each `F{i,j}` denotes the ith block of $\boldsymbol{F}_{j-1}$ ($i = 1, \ldots,$ nBLOCK, $j = 1, \ldots,$mDIM+1). For simplicity, nBLOCK is denoted as n. In Example 1, we have

```
>> F = cell(1,4);
>> F{1,1} = [-11 0 ; 0 23];
           :
>> F{1,4} = [0 -8 ; -8 2];
>> F
F =

    [2x2 double] [2x2 double] [2x2 double] [2x2 double]
```

and in Example 2

```
>> F = cell(3,6);
>> F{1,1} = [-1.4 -3.2 ; -3.2 -28];
          :
>> F{3,6} = [6.1 ; -1.5];
>> F
F =

  Columns 1 through 4

    [2x2 double] [2x2 double] [2x2 double] [2x2 double]
    [3x3 double] [3x3 double] [3x3 double] [3x3 double]
    [2x1 double] [2x1 double] [2x1 double] [2x1 double]

  Columns 5 through 6

    [2x2 double] [2x2 double]
    [3x3 double] [3x3 double]
    [2x1 double] [2x1 double]
```

Note that the diagonal blocks, the third block of each column in the above, are expressed as vectors. Sparse matrices are also available as input. In this case, input can be only the upper (or lower) triangular part of the block. In Example 2,

```
>> F{1,1}

ans =

   (1,1)       -1.4000
   (1,2)       -3.2000
   (2,2)      -28.0000
```

is the correct input of $(1,1)$th element of $\boldsymbol{F}$. Diagonal blocks are expressed as symmetric matrices like non-diagonal blocks. Dense matrices and sparse matrices can be used simultaneously.

```
>> F

F =

     [2x2 sparse] [2x2 double] [2x2 double] [2x2 double]
```

is legitimate as input $\boldsymbol{F}$ of Example 1. At the end, the empty cell is assumed to be a zero matrix.

5.2. File Input

As we have seen in Section 4., we can input data of an SDP from files. Use read_data function to read data files. There are two types of input data file formats. One is the dense format, and the other is the sparse format. The function read_data recognizes the file extension “dat-s” or “dat” automatically. So both,

```
>> [mDIM,nBLOCK,bLOCKsTRUCT,c,F] = read_data('example1.dat');
 and
>> [mDIM,nBLOCK,bLOCKsTRUCT,c,F] = read_data('example1.dat-s');
```

are allowed. Next two sections explain the dense and sparse formats for input data.

5.3. Dense Input Data File

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

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

- In general, the structure of an input data file is as follows:
  Title and Comment
  m — the number of the primal variables x_i's
  nBLOCK — the number of blocks
  bLOCKsTRUCT — the block structure vector
  $\boldsymbol{c}$
  $\boldsymbol{F}_0$
  $\boldsymbol{F}_1$
  ·
  ·
  $\boldsymbol{F}_m$
- Title and Comment — This part is ignored by read data function.
- mDIM — 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.
- nBLOCK,bLOCKsTRUCT — We separately write each of nBLOCK and bLOCKsTRUCT in one line. Any letter after either of nBLOCK and bLOCKsTRUCT through the end of the line is neglected. In addition to blank letter(s), and the tab code(s), we can use the letters

  $$, \quad (\quad) \quad \{ \quad \}$$

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

  ```
  1 = nBLOCK
  2 = bLOCKsTRUCT
  ```

  in Example 1 (see the file "example1.dat" in Section 5.3.), and

```
3 = nBLOCK
2 3 -2 = bLOCKsTRUCT
```

in Example 2 (see the file “example2.dat” in Section 5.3.). In either case, the letters “=nBLOCK” and “= bLOCKsTRUCT” are neglected.

- $\boldsymbol{c}$ — In addition to blank letter(s) and tab code(s), we can use the letters

  $$, \quad (\quad) \quad \{ \quad \}$$

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

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

  in Example 1 (see the file “example1.dat” in Section 5.3.), and

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

  in Example 2 (see the file “example2.dat” in Section 5.3.).

- $\boldsymbol{F}$ — In addition to blank letter(s) and tab code(s), we can use the letters

  $$, \quad (\quad) \quad \{ \quad \}$$

  to separate elements of the matrices $\boldsymbol{F}_0, \boldsymbol{F}_1, \ldots, \boldsymbol{F}_m$ and their elements. In the general case of the block diagonal matrix F given in (1), we write the elements of $\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_l$ sequentially; when $\boldsymbol{B}_i$ is a diagonal matrix, we write only the diagonal element sequentially. If the matrix $\boldsymbol{F}$ is given by (2) (nBLOCK $= 3$, bLOCKsTRUCT $= (3, 2, -2)$), the corresponding representation of the matrix $\boldsymbol{F}$ 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 5.3. 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 }
}
```

Remark. We could 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
```

5.4. Sparse Input File

In the previous subsection, we have stated the dense data format for inputting the data m, n, $\boldsymbol{c} \in R^m$ and $\boldsymbol{F}_i \in \mathcal{S}$ $(i = 0, 1, 2, \ldots, m)$. When not only the constant matrices $\boldsymbol{F}_i \in \mathcal{S}$ $(i = 0, 1, 2, \ldots, m)$ 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.

We show below the file "example1.dat-s", which contains the data of Example 1 (Section 3.2.) 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 5.3. 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". Each line of the rest of the file "example1.dat-s" describes a single element of a constant matrix $\boldsymbol{F}_i$; the 6th line "0 1 1 1 -11" means that the $(1,1)$th element of the 1st block of the matrix $\boldsymbol{F}_0$ 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 $\boldsymbol{F}_3$ is -8.

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

Title and Comment
m — the number of the primal variables x_i's
nBLOCK — the number of blocks
bLOCKsTRUCT — the block structure vector
$\boldsymbol{c}$
$s_1\ b_1\ i_1\ j_1\ v_1$
$s_2\ b_2\ i_2\ j_2\ v_2$
$\cdots$
$s_p\ b_p\ i_p\ j_p\ v_p$
$\cdots$
$s_q\ b_q\ i_q\ j_q\ v_q$

Here $s_p \in \{0,1,\ldots,m\}$, $b_p \in \{1,2,\ldots,\text{nBLOCK}\}$, $1 \le i_p \le j_p$ and $v_p \in R$. 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_pth block of the constant matrix $\boldsymbol{F}_{s_p}$ is v_p. If the b_pth block is an $\ell \times \ell$ symmetric (non-diagonal) matrix then (i_p, j_p) must satisfy $1 \le i_p \le j_p \le \ell$; hence only nonzero elements in the upper triangular part of the b_pth block are described in the file. If the b_pth block is an $\ell \times \ell$ diagonal matrix then (i_p, j_p) must satisfy $1 \le i_p = j_p \le \ell$.

6. Initial Point Input

6.1. Initial Point

If a feasible interior solution $(\boldsymbol{x}^0, \boldsymbol{X}^0, \boldsymbol{Y}^0)$ is known in advance, we may want to start the SDPA-M from $(\boldsymbol{x}^0, \boldsymbol{X}^0, \boldsymbol{Y}^0)$. In such a case, we can optionally specify the feasible-interior solution as initial point. It is often the case that such an initial point is not known or you want to execute without any initial point. In such case an initial point can be omitted as an input argument(see Section 9.).

- $\boldsymbol{x}^0$ — Initial point of $\mathcal{P}$. (mDIM×1) vector.
- $\boldsymbol{X}^0$ — Initial point of $\mathcal{P}$. (nBLOCK×1) cell matrix.
- $\boldsymbol{Y}^0$ — Initial point of $\mathcal{D}$. (nBLOCK×1) cell matrix.

$\boldsymbol{x}^0$ can be a column vector. Like $\boldsymbol{F}$ in Section 5.1., each cell of $\boldsymbol{X}^0$ and $\boldsymbol{Y}^0$ can be dense or sparse. In case of using sparse matrices, input only the upper(or lower) triangular part. Sparse cell and dense cell can be used simultaneously.

6.2. File Input

We can import an initial point from a file by using initiall_point function. An initial point file also has dense or sparse format. As the initial_point function reads an initial point from a file, it recognizes the file extension "ini" and "ini-s". Both

```
>> [x0,X0,Y0] = initial_point('example1.ini',mDIM,nBLOCK,bLOCKsTRUCT);
  and
>> [x0,X0,Y0] = initial_point('example1.ini-s',mDIM,nBLOCK,bLOCKsTRUCT);
```

are allowed.

6.3. Dense Initial Point File

In general, an initial point file can have any name with the postfix ".ini" or ".ini-s"; for example, "example.ini" is a legitimate initial point filename.

An initial point file contains the data

$\boldsymbol{x}^0$
$\boldsymbol{X}^0$
$\boldsymbol{Y}^0$

in this order, where the description of the m-dimensional vector $\boldsymbol{x}^0$ must follow the same format as the constant vector $\boldsymbol{c}$ (see Section 5.3.), and the description of $\boldsymbol{X}^0$ and $\boldsymbol{Y}^0$, the same format as the constraint matrix $\boldsymbol{F}_i$ (see Section 5.3.).

We show below the file "example1.ini", which contains an initial point data of Example 1 in the dense data format.

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

6.4. 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 6.3. 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 $\boldsymbol{x}^0$ in the dense format. Each line of the rest of the file "example1.ini-s" describes a single element of an initial matrix $\boldsymbol{X}^0$ if the first number of the line is 1 or a single element of an initial matrix $\boldsymbol{Y}^0$ 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 $\boldsymbol{X}^0$ 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 $\boldsymbol{Y}^0$ 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 initial point data files. The SDPA-M distinguishes a sparse initial point data file with the postfix ".ini" from a dense initial point data file with the postfix ".ini-s".

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

$\boldsymbol{x}^0$
$s_1\ b_1\ i_1\ j_1\ v_1$
$s_2\ b_2\ i_2\ j_2\ v_2$
$\quad\cdots$
$s_p\ b_p\ i_p\ j_p\ v_p$
$\quad\cdots$
$s_q\ b_q\ i_q,\ j_q\ v_q$

Here $s_p = 1$ or 2, $b_p \in \{1, 2, \ldots, \text{nBLOCK}\}$, $1 \leq i_p \leq j_p$ and $v_p \in R$. 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_pth block of the constant matrix $\boldsymbol{X}^0$ 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_pth block of the constant matrix $\boldsymbol{Y}^0$ is v_p. If the b_pth 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_pth block are described in the file. If the b_pth block is an $\ell \times \ell$ diagonal matrix then (i_p, j_p) must satisfy $1 \leq i_p = j_p \leq \ell$.

7. Optional Setting

7.1. Option Value

We can specify various parameters before an execution of SDPA-M. As its name indicates, OPTION can be omitted as input argument (see Section 9.). In that case the default values are used. First we show the default option values below. Then each element is explained.

```
OPTION =

    maxIteration: 40
     epsilonStar: 1.0000e-07
      lambdaStar: 100
       omegaStar: 2
      lowerBound: -100000
      upperBound: 100000
        betaStar: 0.1000
         betaBar: 0.2000
       gammaStar: 0.9000
     epsilonDash: 1.0000e-07
     isSymmetric: 0
           print: 'display'
```

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

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

  the SDPA-M stops. Too small epsilonStar and epsilonDash may cause a numerical instability. A reasonable choice is epsilonStar $\geq$ 1.0E-7.
- lambdaStar — This parameter determines an initial point $(\boldsymbol{x}^0, \boldsymbol{X}^0, \boldsymbol{Y}^0)$ such that

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

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

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

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

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

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-M searches a maximum solution $\boldsymbol{Y}$ within the region

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

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 larger lambdaStar and a smaller omegaStar > 1.

- lowerBound — Lower bound of the minimum objective value of the primal problem $\mathcal{P}$. When the SDPA-M generates a primal feasible solution $(\boldsymbol{x}^k, \boldsymbol{X}^k)$ whose objective value $\sum_{i=1}^{m} c_i x_i^k$ gets smaller than the lowerBound, the SDPA-M 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-M generates a dual feasible solution $\boldsymbol{Y}^k$ whose objective value $\boldsymbol{F}_0 \bullet \boldsymbol{Y}^k$ gets larger than the upperBound, the SDPA-M 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 — A parameter controlling the search direction when $(\boldsymbol{x}^k, \boldsymbol{X}^k, \boldsymbol{Y}^k)$ is feasible. As we take a smaller betaStar > 0.0, the search direction can get close to the affine scaling direction without centering.

- betaBar — A parameter controlling the search direction when $(\boldsymbol{x}^k, \boldsymbol{X}^k, \boldsymbol{Y}^k)$ is infeasible. As we take a smaller betaBar > 0.0, the search direction can get close to the affine scaling direction without centering. The value of betaBar must be not less than the value of betaStar; $0 \leq$ betaStar $\leq$ betaBar.

- gammaStar — A reduction factor for the primal and dual step lengths; $0.0 <$ gammaStar $<$ 1.0.

- isSymmetric — a 0-1 flag indicating whether to check the symmetricity of input matrices.

- print — Destination of file output.

```
>> OPTION.print = 'display'  to have file output on display
                  'filename' to have file output on the file named 'filename'
                  'no'       to have no file output
```

7.2. More on Option Value

We may encounter some numerical difficulty during the execution of the SDPA-M with the default option values set by param.m, 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 option values: betaStar, betaBar, and gammaStar. We present below two sets of those option values. 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

```
OPTION.betaStar = 0.1 ;
OPTION.betaBar = 0.2 ;
OPTION.gammaStar = 0.9 ;
```

Unstable but Fast

```
OPTION.betaStar = 0.01 ;
OPTION.betaBar = 0.02 ;
OPTION.gammaStar = 0.98 ;
```

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

8. Output

8.1. Output Arguments

The function sdpam has five output arguments.

- objVal — Objective function value.objVal is a (1×2) matrix such as [objValP objValD]. They are objective function value of a standard form SDP problem $\mathcal{P}$ and $\mathcal{D}$ (see Section 3.1.) respectively.
- $\boldsymbol{x}$ — Solution of $\mathcal{P}$; (mDIM $\times$ 1) vector.
- $\boldsymbol{X}$ — Solution of $\mathcal{P}$; (nBLOCK $\times$ 1) cell matrix.
- $\boldsymbol{Y}$ — Solution of $\mathcal{D}$; (nBLOCK $\times$ 1) cell matrix.
- INFO — Information about the execution result. INFO is a structure including the information noted below.

phasevalue	the status when the iteration stops
iteration	the number of iterations
cputime	the total CPU time(seconds)

  The value which phasevalue takes is explained in the next section.

8.2. File Output

By defaults SDPA-M shows some information on the display. In the case of Example 1, we have

```
SDPA library start ... (built at May 23 2005 14:54:57)

   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 1.0e-16 9.4e-02 +8.39e+02 +7.51e+01 2.3e+00 9.6e-01 2.00e-01
 2 1.7e+02 1.9e-16 3.6e-03 +1.96e+02 -3.74e+01 1.3e+00 1.0e+00 2.00e-01
 3 1.8e+01 1.6e-16 2.2e-17 -6.84e+00 -4.19e+01 9.9e-01 9.9e-01 1.00e-01
 4 1.9e+00 1.6e-16 1.5e-17 -3.81e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01
 5 1.9e-01 1.6e-16 1.5e-17 -4.15e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01
 6 1.9e-02 1.6e-16 7.5e-18 -4.19e+01 -4.19e+01 1.0e-00 9.0e+01 1.00e-01
 7 1.9e-03 1.6e-16 5.8e-16 -4.19e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01
 8 1.9e-04 1.8e-16 2.2e-17 -4.19e+01 -4.19e+01 1.0e-00 9.0e+01 1.00e-01
 9 1.9e-05 1.5e-16 3.1e-15 -4.19e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01
10 1.9e-06 1.6e-16 2.2e-17 -4.19e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01

phase.value = pdOPT
   Iteration = 10
          mu = 1.9180668442025276e-06
relative gap = 9.1554506595324411e-08
         gap = 3.8361336884050552e-06
      digits = 7.0383202735216370e+00
objValPrimal = -4.1899996163866327e+01
objValDual   = -4.1899999999999977e+01
p.feas.error = 1.9317880628477724e-14
d.feas.error = 2.1316282072803006e-14
```

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

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

  holds.

- thetaP — The SDPA-M starts with thetaP = 0.0 if the initial point $(\boldsymbol{x}^0, \boldsymbol{X}^0)$ 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 kth iteration is given by

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

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

- thetaD — The SDPA-M starts with thetaD = 0.0 if the initial point $\boldsymbol{Y}^0$ 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 kth iteration is given by

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

  The thetaD is theoretically monotone nonincreasing, and when it gets 0.0, we obtain a dual feasible solution $\boldsymbol{Y}^k$. 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 $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y})$ of the SDP such that

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

  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 $\boldsymbol{Y}$ such that

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

  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 $(\boldsymbol{x}, \boldsymbol{X})$ such that

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

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

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

- Iteration — The iteration number which the SDPA-M needs to terminate.
- relative gap — The relative gap means that

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

  This value is compared with **epsilonStar** (Section 7.).
- gap — The gap means that mu $\times$ n.
- digits — This value indicates how objP and objD resemble by the following definition.

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

- objValPrimal — The primal objective function value.

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

- objValDual — The dual objective function value.

$$\mathrm{objValDual} = \boldsymbol{F}_0 \bullet \boldsymbol{Y}.$$

- p.feas.error — This value is the primal infeasibily in the last iteration,

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

  This value is compared with **epsilonDash** (Section 7.). Even if primal is feasible, this value may not be 0 because of numerical error.
- d.feas.error — This value is the dual infeasibily in the last iteration,

$$\mathrm{d.feas.error} = \max\{|\boldsymbol{F}_i \bullet \boldsymbol{Y} - c_i| \ : i = 1, 2, \ldots, m\}.$$

  This value is compared with **epsilonDash** (Section 7.). Even if dual is feasible, this value may not be 0 because of numerical error.

8.3. Error Message

During a computation if there is a message such as:

- “Step length is too small.” or
- “cholesky miss condition :: not positive definite”,

changing the current maxIteration for a large number does not make the result better. In the first case, you have to make gammaStar smaller. In the other case, you have to change betaStar, betaBar, gammaStar by try and error, but it is possible that the problem itself is hard to compute by this algorithm.

9. Sample Runs

The function sdpam can be overloaded as shown below.

```
>> [objVal,x,X,Y,INFO] = sdpam(mDIM,nBLOCK,bLOCKsTRUCT,c,F)
>> [objVal,x,X,Y,INFO] = sdpam(mDIM,nBLOCK,bLOCKsTRUCT,c,F,OPTION)
>> [objVal,x,X,Y,INFO] = sdpam(mDIM,nBLOCK,bLOCKsTRUCT,c,F,x0,X0,Y0)
>> [objVal,x,X,Y,INFO] = sdpam(mDIM,nBLOCK,bLOCKsTRUCT,c,F,x0,X0,Y0,OPTION)
```

If you want to specify the initial point , include $\boldsymbol{x}^0, \boldsymbol{X}^0, \boldsymbol{Y}^0$ as input argument.

```
>> [objVal,x,X,Y,INFO] = sdpam(mDIM,nBLOCK,bLOCKsTRUCT,c,F,x0,X0,Y0)
 or
>> [objVal,x,X,Y,INFO] = sdpam(mDIM,nBLOCK,bLOCKsTRUCT,c,F,x0,X0,Y0,OPTION)
```

Otherwise initial point is calculated from parameter lambdaStar. If you want to modify `OPTION`, make sure to include `OPTION` as input argument.

```
>> [objVal,x,X,Y,INFO] = sdpam(mDIM,nBLOCK,bLOCKsTRUCT,c,F,OPTION)
 or
>> [objVal,x,X,Y,INFO] = sdpam(mDIM,nBLOCK,bLOCKsTRUCT,c,F,x0,X0,Y0,OPTION)
```

Otherwise the default values take precedence even if you modify `OPTION` value.

10. SDP Feasibility Problem

10.1. Conversion into Standard Form SDP

In this section, we explain how to solve an SDP feasibility problem, i.e., in the case of $\boldsymbol{c} = \boldsymbol{0}$.

First of all, the SDP feasibility problem is defined by

$$\text{find } \boldsymbol{x} \in R^m \quad \text{such that} \quad \boldsymbol{X} = \sum_{i=1}^{m} \boldsymbol{F}_i x_i - \boldsymbol{F}_0, \quad \mathcal{S} \ni \boldsymbol{X} \succeq \boldsymbol{O}, \tag{3}$$

where $\boldsymbol{x} = [x_1, x_2, \cdots, x_m]$ and $x_i \in R$ for all i. This problem can be converted into the standard form SDP by introducing a slack variable t such as

$$\text{minimize } t \quad \text{subject to} \quad \boldsymbol{X} = \sum_{i=1}^{m} \boldsymbol{F}_i x_i - \boldsymbol{F}_0 + t\boldsymbol{I}, \quad \mathcal{S} \ni \boldsymbol{X} \succeq \boldsymbol{O}. \tag{4}$$

Therefore, $\boldsymbol{x}$ and $\boldsymbol{c}$ become

$$\boldsymbol{x} := [x_1, \cdots, x_m, t]^T, \quad \boldsymbol{c} := [0, \cdots, 0, 1]^T.$$

By solving the problem (4), if we obtain $t < 0$, then we conclude that the problem is feasible.

But to solve the problem (4) by SDPA-M, we have to add one more constraint which gives a lower bound of t otherwise, we will find a numerical problem. For choosing the lower bound of

t, the most natural one is $t > -1$. Although changing the lower bound of t makes the solution different, the solution itself is not important as long as a sign of the solution t doesn't change.

Therefore, in practice instead of solving the problem (3) directly, we solve the problem

$$\text{minimize } \ t \quad \text{subject to} \quad \boldsymbol{X} = \sum_{i=1}^{m} \boldsymbol{F}_i x_i - \boldsymbol{F}_0 + t\boldsymbol{I}, \ \ \mathcal{S} \ni \boldsymbol{X} \succeq \boldsymbol{O}, \ \ t > -1. \tag{5}$$

10.2. Example

Let us consider the following feasibility problem of the linear matrix inequalities:

$$\text{find } \boldsymbol{X} \in R^{2\times 2} \quad \text{such that} \quad \boldsymbol{X}\boldsymbol{A} + \boldsymbol{A}^T\boldsymbol{X} < \boldsymbol{O}, \ \ \boldsymbol{X} > \boldsymbol{O}, \tag{6}$$

where $\boldsymbol{A} \in R^{2\times 2}$ is given. It is known that if the real part of all eigenvalues of $\boldsymbol{A}$ are negative, then there exists $\boldsymbol{X}$; this is nothing but a Lyapunov stability condition. To avoid the numerical problem, we replace $\boldsymbol{X} > 0$ with $\boldsymbol{X} > \boldsymbol{I}$ in (6). Then, the problem (6) is converted into

$$\begin{aligned} &\text{minimize } \ \boldsymbol{c}^T\boldsymbol{x} \quad \text{subject to} \\ &\boldsymbol{F}_1 x_1 + \boldsymbol{F}_2 x_2 + \boldsymbol{F}_3 x_3 + \boldsymbol{F}_4 t - \boldsymbol{F}_0 \succeq \boldsymbol{O}, \end{aligned} \tag{7}$$

where

$$\begin{bmatrix} x_1 & x_2 \\ x_2 & x_3 \end{bmatrix} := \boldsymbol{X}, \quad \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} := \boldsymbol{A}, \quad \boldsymbol{c} := \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T, \quad \boldsymbol{x} := \begin{bmatrix} x_1 & x_2 & x_3 & t \end{bmatrix}^T,$$

$$\boldsymbol{F}_1 := \left[\begin{array}{cc|cc|c} -2a_{11} & -a_{12} & 0 & 0 & 0 \\ -a_{12} & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right], \quad \boldsymbol{F}_2 := \left[\begin{array}{cc|cc|c} -2a_{21} & -a_{11}-a_{22} & 0 & 0 & 0 \\ -a_{11}-a_{22} & -2a_{12} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right],$$

$$\boldsymbol{F}_3 := \left[\begin{array}{cc|cc|c} 0 & -a_{21} & 0 & 0 & 0 \\ -a_{21} & -2a_{22} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right], \quad \boldsymbol{F}_4 := \left[\begin{array}{cc|cc|c} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right], \quad \boldsymbol{F}_0 := \left[\begin{array}{cc|cc|c} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & -1 \end{array}\right].$$

Note that the 1st diagonal block of $\boldsymbol{F}$ matrices corresponds with $\boldsymbol{X}\boldsymbol{A} + \boldsymbol{A}^T\boldsymbol{X} < t\boldsymbol{I}$, the 2nd diagonal block corresponds with $\boldsymbol{X} - \boldsymbol{I} > -t\boldsymbol{I}$, and the 3rd diagonal block corresponds with $t > -1$, respectively.

Let us solve the case

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ -3 & -4 \end{bmatrix},$$

that is, all eigenvalues are negative, $\lambda(\boldsymbol{A}) = -1, -3$. To solve this problem, we write the following MATLAB code:

```
A=[0 1; -3 -4];
mDIM=4;
```

```
nBLOCK=3;
bLOCKsTRUCT=[2 2 -1];
c=[0 0 0 1];
F=cell(3,5);
% F_{0}
F{2,1}=eye(2);
F{3,1}=-1;
% F_{1}
F{1,2}=[-2*A(1,1) -A(1,2); -A(1,2) 0];
F{2,2}=[1 0; 0 0];
% F_{2}
F{1,3}=[-2*A(2,1) -A(1,1)-A(2,2); -A(1,1)-A(2,2) -2*A(1,2)];
F{2,3}=[0 1; 1 0];
% F_{3}
F{1,4}=[0 -A(2,1); -A(2,1) -2*A(2,2)];
F{2,4}=[0 0; 0 1];
% F_{4}
F{1,5}=eye(2,2);
F{2,5}=eye(2,2);
F{3,5}=1;
% solve the problem by SDPA-M
[objVal,x,X,Y,INFO]=sdpam(mDIM,nBLOCK,bLOCKsTRUCT,c,F);
```

The output from SDPA-M is as follows:

```
SDPA library start ... (built at May 23 2005 14:54:57)

   mu      thetaP  thetaD  objP      objD      alphaP  alphaD  beta
 0 1.0e+04 1.0e+00 1.0e+00 -0.00e+00 +1.00e+02 1.0e+00 9.0e-01 2.00e-01
 1 1.4e+03 2.8e-16 1.0e-01 +1.39e+02 +9.60e+00 9.0e-01 9.0e-01 2.00e-01
 2 2.3e+02 2.8e-16 1.1e-02 +1.73e+02 +9.19e-01 2.6e+00 8.6e-01 2.00e-01
 3 4.6e+01 2.8e-16 1.5e-03 +1.64e+01 -4.21e-02 1.9e+00 9.0e-01 2.00e-01
 4 7.8e+00 5.6e-16 1.6e-04 +7.37e-01 -8.98e-01 9.3e+00 8.9e-01 2.00e-01
 5 3.5e+00 7.0e-15 1.7e-05 -8.26e-01 -9.87e-01 9.0e-01 9.0e-01 2.00e-01
 6 5.8e-01 9.8e-15 1.8e-06 -3.62e-01 -9.99e-01 1.1e+00 9.0e-01 2.00e-01
 7 7.8e-02 9.1e-15 1.8e-07 -9.36e-01 -1.00e-00 1.2e+00 9.0e-01 2.00e-01
 8 1.1e-02 1.4e-14 1.8e-08 -9.94e-01 -1.00e-00 1.4e+00 9.0e-01 2.00e-01
 9 1.6e-03 1.7e-14 1.8e-09 -9.99e-01 -1.00e-00 1.8e+00 9.0e-01 2.00e-01
10 2.4e-04 1.8e-14 1.8e-10 -1.00e-00 -1.00e-00 3.8e+00 9.0e-01 2.00e-01
11 5.4e-05 5.6e-14 1.8e-11 -1.00e-00 -1.00e-00 6.0e+00 9.0e-01 1.00e-01
12 8.0e-06 7.2e-14 1.8e-12 -1.00e-00 -1.00e-00 8.9e-01 8.9e-01 1.00e-01
13 1.4e-06 1.2e-14 1.9e-13 -1.00e-00 -1.00e-00 1.1e+00 9.0e-01 1.00e-01
14 2.0e-07 7.3e-14 2.0e-14 -1.00e-00 -1.00e-00 1.1e+00 9.0e-01 1.00e-01

phase.value = pdOPT
   Iteration = 14
          mu = 1.9969468324606384e-07
relative gap = 7.8366267142193635e-08
```

```
          gap = 9.9847341623031924e-07
       digits = 7.1058708226647997e+00
objValPrimal = -9.9999992160167883e-01
objValDual   = -9.9999999996794597e-01
p.feas.error = 7.3980821468921931e-12
d.feas.error = 1.7688923188844278e-11
```

Since $t < 0$ (see a value of `objValPrimal` in the above result), we conclude that the original problem is feasible. On the other hand, if we choose

$$A = \begin{bmatrix} 0 & 1 \\ -3 & 4 \end{bmatrix},$$

which has positive eigenvalues, $\lambda(\boldsymbol{A}) = 1, 3$, we obtain

```
SDPA library start ... (built at May 23 2005 14:54:57)

   mu      thetaP  thetaD  objP      objD      alphaP  alphaD  beta
 0 1.0e+04 1.0e+00 1.0e+00 -0.00e+00 +1.00e+02 1.0e+00 9.0e-01 2.00e-01
 1 1.4e+03 1.4e-19 1.0e-01 +1.40e+02 +9.80e+00 9.0e-01 9.0e-01 2.00e-01
 2 2.1e+02 2.8e-16 9.7e-03 +1.84e+02 +1.21e+00 9.3e-01 9.3e-01 2.00e-01
 3 4.9e+01 5.6e-16 6.4e-04 +1.84e+02 +3.07e-01 1.1e+00 1.0e+00 2.00e-01
 4 8.5e+00 8.4e-16 1.6e-19 +4.26e+01 +2.70e-01 9.6e-01 1.4e+01 1.00e-01
 5 1.1e+00 8.0e-16 1.0e-17 +6.11e+00 +6.33e-01 9.5e-01 9.7e+00 1.00e-01
 6 1.4e-01 8.0e-16 8.3e-17 +1.45e+00 +7.23e-01 9.6e-01 1.0e+00 1.00e-01
 7 1.9e-02 8.1e-16 1.1e-18 +9.41e-01 +8.47e-01 9.5e-01 9.8e-01 1.00e-01
 8 2.5e-03 8.1e-16 3.2e-19 +8.86e-01 +8.73e-01 9.4e-01 1.0e+00 1.00e-01
 9 3.3e-04 8.1e-16 1.7e-18 +8.79e-01 +8.77e-01 9.7e-01 1.1e+00 1.00e-01
10 3.6e-05 8.1e-16 2.7e-18 +8.78e-01 +8.77e-01 9.7e-01 1.1e+00 1.00e-01
11 3.8e-06 8.1e-16 1.1e-18 +8.77e-01 +8.77e-01 9.7e-01 1.1e+00 1.00e-01
12 3.9e-07 8.1e-16 1.3e-18 +8.77e-01 +8.77e-01 9.8e-01 1.1e+00 1.00e-01
13 3.9e-08 8.1e-16 4.3e-18 +8.77e-01 +8.77e-01 9.8e-01 1.1e+00 1.00e-01
14 3.9e-09 8.1e-16 2.9e-18 +8.77e-01 +8.77e-01 9.8e-01 1.1e+00 1.00e-01

phase.value = pdOPT
   Iteration = 14
          mu = 3.9277516936612115e-09
relative gap = 1.9638726311832500e-08
          gap = 1.9638758468306057e-08
       digits = 7.6501264737392791e+00
objValPrimal = 8.7748519224301846e-01
objValDual   = 8.7748517260429215e-01
p.feas.error = 8.1712414612411521e-14
d.feas.error = 1.9984014443252818e-15
```

Since $t > 0$, we conclude that this problem is not feasible as we expected.

References

[1] M. Kojima K. Fujisawa, K. Nakata and M. Yamashita. SDPA(SemiDefinite Programming Algorithm) User's Mannual — Version 6.20. Research report, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2-12-1-W8-29 Oh-Okayama, Meguro-ku, Tokyo 152-8552, Japan, 2005. Available via the WWW site at http://grid.r.dendai.ac.jp/sdpa/.

[2] J. Löfberg. YALMIP : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. Available from `http://control.ee.ethz.ch/~joloef/yalmip.php`.

[3] M.J. Todd. Semidefinite optimization. *Acta Numerica*, 10:515–560, 2001.

[4] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1996.