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

Tokyo Institute of Technology
2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-0033, Japan

SDPA-M (SemiDefinite Programming Algorithm in MATLAB) User's Manual — Version 2.00

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

B-359, January 2000
Revised: June 2003

Abstract. The SDPA-M (Semidfinite Programming Algorithm in MATLAB) Version 2.0 is a MATLAB interface of the SDPA Version 6.0 [2], which is known as a fast and numerically stable solver for SDPs (semidefinite programs) [4, 5]. 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, the user can easily manipulate and transform his own problems in the MATLAB language, and then solve them by the SDPA-M. This manual and the SDPA-M can be found in 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:nihou@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. Installation

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

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

The SDPA-M requires the SDPA Verion 6.0 [2], so that you have to install the SDPA Verion 6.0 as a library before installing this package. The latest version, the Version 2.00 of this package is just different from the previous version, the Version 1.00 in the use of different versions of the SDPA; the previous Version 1.00 uses the SDPA Version 5.0 with Meschach[3] for matrix computation while the latest version uses the SDPA Version 6.0 with ATLAS and LAPACK[1] for matrix computation. In general, the latest version solves SDPs faster than the previous version.

First move the downloaded file (sdpam.2.00.tar.gz) into a suitable directory. To resolve this file, execute the following procedures:

tar xvzf sdpam.2.00.tar.gz

The tar command will create the subdirectory **sdpam** in which you can find the following directories and files:

Makefile		An Makefile for this package.
sdpam.m		An M-file to solve the SDPs.
read data.m		An M-file to read SDPA format data file.
initial_point.m		An M-file to read SDPA format initial point file.
param.m		An M-file to set default SDPA-M parameters.
mex/	**Makefile**	A makefile for creating a MEX-file.
	mexsdpa.cpp	A C MEX-file coded with C++ for the gateway.
	installmex.sh	A shell script for installing a MEX-file.
examples/	**example1.dat**	A sample input file No.1 in the dense data format.
	example2.dat	A sample input file No.2 in the dense data format.
	example1.dat-s	A sample input file in the sparse data format.
	example1.ini	An initial point file in the dense data format.
	example1.ini-s	An initial point file in the sparse data format.
doc/	**sdpamManual.ps**	This user's manual.

For installing this package, you have to change two Makefiles for your environment. At the top directory, you'll find "Makefile" and you have to change a line of the file as follows:

```
line 16 :: INSTALL_DIR=$(HOME)/matlab/sdpam
```

This "line 16" specifies an install target directory. Thus, if you want to install this package to a directory "/opt/matlab/sdpam", you have to change this line to

```
INSTALL_DIR=/opt/matlab/sdpam
```

Next change directory to the "mex" directory by typing "cd mex" to find the other "Makfile"; then you have to modify four lines as follows:

```
line  5 :: LAPACK=$(HOME)/lapack
line  6 :: SDPA=$(HOME)/sdpa
line  7 :: MEX=/usr/local/bin/mex
line  9 :: MATLABV=-D_MATLAB_V6_5_
```

"line 5" and "line 6" specify paths for LAPACK and SDPA, respectively, and "line 7" a path for the "mex" command. "line 9" denotes a command switch for the "mex" command. If your MATLAB version is lower than 6.5, you have to comment out this line as follows:

```
#MATLABV=-D_MATLAB_V6_5_
```

So far you have changed two Makefiles. Then the next step is creating a MEX-file and installing this package. If you want to install solver files and File I/O M-files, type as follows:

```
$ make; make install.compact
```

If you want to install only solver files (in this case, File I/O M-files are not installed), type as follows:

```
$ make; make install.minimum
```

After installing the files, we recommend you to add this package path to the MATLAB search path by adding the following line to "$HOME/matlab/startup.m."

```
addpath <INSTALL_DIR>
```

Here "<INSTALL_DIR>" means the path which you installed this package. For examples, if you install this package to "/home/user1/matlab/sdpam", add a line to "$HOME/matlab/startup.m"

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

Now you have completed the installation of this package. Please check whether the installation is finished successfully by solving a simple SDP (See Section 3.).

2. Semidefinite Program

2.1. Standard Form SDP and Its Dual

The SDPA-M solves the following standard form semidefinite program [4, 5] and its dual. Here

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

$\mathcal{S}$: 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.*, $\sum_{i=1}^{n}\sum_{j=1}^{n} U_{ij}V_{ij}$

$\boldsymbol{U} \succeq \boldsymbol{O}$, $\Longleftrightarrow$ $\boldsymbol{U} \in \mathcal{S}$ 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), 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

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

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

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

2.2. 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

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

The data (see Section 4.3.) of this problem is contained in the file "example1.dat".

2.3. Example 2

$$
\begin{aligned}
m &= 5, \ n = 7, \ \boldsymbol{c} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix} = \begin{pmatrix} 1.1 \\ -10 \\ 6.6 \\ 19 \\ 4.1 \end{pmatrix}, \\
\boldsymbol{F}_0 &= \begin{pmatrix}
-1.4 & -3.2 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
-3.2 & -28 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 15 & -12 & 2.1 & 0.0 & 0.0 \\
0.0 & 0.0 & -12 & 16 & -3.8 & 0.0 & 0.0 \\
0.0 & 0.0 & 2.1 & -3.8 & 15 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.8 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -4.0
\end{pmatrix}, \\
\boldsymbol{F}_1 &= \begin{pmatrix}
0.5 & 5.2 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
5.2 & -5.3 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 7.8 & -2.4 & 6.0 & 0.0 & 0.0 \\
0.0 & 0.0 & -2.4 & 4.2 & 6.5 & 0.0 & 0.0 \\
0.0 & 0.0 & 6.0 & 6.5 & 2.1 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -4.5 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -3.5
\end{pmatrix}
\end{aligned}
$$

$$
\begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array}
$$

$$\boldsymbol{F}_5 = \begin{pmatrix} -6.5 & -5.4 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -5.4 & -6.6 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 6.7 & -7.2 & -3.6 & 0.0 & 0.0 \\ 0.0 & 0.0 & -7.2 & 7.3 & -3.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -3.6 & -3.0 & -1.4 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 6.1 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -1.5 \end{pmatrix}.$$

As shown in this example, the SDPA-M handles block diagonal matrices. The data (see Section 4.3.) of this example is contained in the file "example2.dat".

3. 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.

```
>> %Get the problem data from data file 'example1.dat'.
>> [mDIM,nBLOCK,bLOCKsTRUCT,c,F] = read_data('example1.dat');
>>
>> %Get initial point from sdpa format initial point file.
>> [x0,X0,Y0] = initial_point('example1.ini',mDIM,nBLOCK,bLOCKsTRUCT);
>>
>> %Next command is to have file output to the file named 'example1.out'.
>> %File output is to be on display by default.
>> OPTION.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 output file named 'example1.out'. To see the file "example1.out"

```
>> type example1.out

SDPA library start ... (built at May 13 2003 10:53:25)
let me see your ...
   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 chapters, each arguments and contents of file output are explained.

4. Problem Data Input

4.1. Input Arguments

- mDIM — Number of primal variable. We have

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

  in Example1, and

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

  in Example2.

- nBLOCK,bLOCKsTRUCT — The number of blocks, the block structure vector. The SDPA-M handles block diagonal matrices as we have seen in Section 3.. In terms of the number of blocks, denoted by nBLOCK, and the block structure vector, denoted by bLOCKsTRUCT, we express a common matrix data structure for the constraint matrices $F_0, F_1, \ldots, F_m$. If we deal with a block diagonal matrix 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:

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

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

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

we have

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

If

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

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

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

We have

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

in Example 1 , and

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

in Example 2. bLOCKsTRUCT can be column vector.

- $\boldsymbol{c}$ — Constant vector. We write all the elements $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_m$ 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}	$\cdots$	$\cdots$	F{1,m+1}
2nd block	F{2,1}	F{2,2}	$\cdots$	$\cdots$	F{2,m+1}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n-th block	F{n,1}	F{n,2}	$\cdots$	$\cdots$	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 only the upper (or lower) triangular part of the block. In Example2,

```
>> F{1,1}

ans =

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

is correct input as $(1,1)$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 F of Example 1.

4.2. File Input

As we have seen in Section 3., 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 format. One is dense format, and the other is 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.

4.3. Dense Input Data File

- "example1.dat" — Input Data File of Example1

```
"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 Example2

```
*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 $\boldsymbol{x}_i$'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

  , ( ) { }

  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 4.3.), and

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

in Example 2 (see the file "example2.dat" in Section 4.3.). 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

  , ( ) { }

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

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

  in Example 1 (see the file "example1.dat" in Section 4.3.), and

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

  in Example 2 (see the file "example2.dat" in Section 4.3.).

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

  , ( ) { }

  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 4.3.. 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
```

4.4. 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 date 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 2.2. 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
$\boldsymbol{c}$
$k_1\ b_1\ i_1\ j_1\ v_1$
$k_2\ b_2\ i_2\ j_2\ v_2$
$\quad\ \ldots$
$k_p\ b_p\ i_p\ j_p\ v_p$
$\quad\ \ldots$
$k_q\ b_q\ i_q\ j_q\ v_q$

Here $k_p \in \{0, 1, \ldots, m\}$, $b_p \in \{1, 2, \ldots, \text{nBLOCK}\}$, $1 \leq i_p \leq j_p$ and $v_p \in R$. 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_pth block of the constant matrix $\boldsymbol{F}_{k_p}$ is v_p. If the b_pth block is an $\ell \times \ell$ symmetric (non-diagonal) matrix then (i_p, j_p) must satisfy $1 \leq i_p \leq j_p \leq \ell$; 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 $1 \leq i_p = j_p \leq \ell$.

5. Initial Point Input

5.1. 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 8.).

- $\boldsymbol{x}^0$ — 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 column vector. Like F in section 4.1, 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.

5.2. File Input

We can import an initial point from a file by using initial—_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.

5.3. 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 4.3.), and the description of $\boldsymbol{X}^0$ and $\boldsymbol{Y}^0$,the same format as the constraint matrix $\boldsymbol{F}_i$(see Section 4.3.).

5.4. 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 5.1. 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 $\boldsymbol{x}^0$ 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:

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

Here $s_p = 1$ or 2, $b_p \in \{1, 2, \ldots, \text{nBLOCK}\}$, $1 \le i_p \le j_p$ 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 $1 \le i_p \le j_p \le \ell$; 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. Optional Setting

6.1. 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 8.). 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

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

  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 strongly recommend to take a sufficiently large lambdaStar 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
```

6.2. 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, gammaStar and deltaStar. 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.

7. Output

7.1. Output Arguments

Function sdpam has five output arguments.

- objVal — Objective function value.objVal is (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 2.1.) respectively.
- $\boldsymbol{x}$ — Solution of $\mathcal{P}$; (mDIM × 1) vector.
- $\boldsymbol{X}$ — Solution of $\mathcal{P}$; (nBLOCK × 1) cell matrix.
- $\boldsymbol{Y}$ — Solution of $\mathcal{D}$; (nBLOCK × 1) cell matrix.
- INFO — Information about the execution result. INFO is structure including the informations noted below.

phasevalue	indicates how the execution stopped
iteration	the number of iterations
cputime	the total CPU time(seconds)

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

7.2. 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 13 2003 10:53:25)
let me see your ...
   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

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

  This value is compared with **epsilonStar** (Section 6.).

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

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

- objValPrimal — The primal objective function value.

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

- objValDual — The dual objective function value.

$$\text{objValD} = \boldsymbol{F}_0 \bullet \boldsymbol{Y}.$$

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

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

  This value is compared with **epsilonDash** (Section 6.). 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,

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

  This value is compared with **epsilonDash** (Section 6.). Even if dual is feasible, this value may not be 0 because of numerical error.

8. Sample Runs

The function sdpam can be overloaded as shown below.

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

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

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

References

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorense. *LAPACK User's Guide Third*. Society for Industrial and Applied Mathematics, Philadelphia,PA, 1999. ISBN 0-89871-447-8 (paperback).

[2] M. Kojima K. Fujisawa, K. Nakata and M. Yamashita. SDPA(SemiDefinite Programming Algorithm) User's Mannual — Version 6.00. Research report, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-0033, Japan, 2002. Available via the WWW site at http://grid.r.dendai.ac.jp/sdpa/.

[3] D. E. Stewart and Meschach Z. leyd. Matrix computation in c. In *Proceedings of the Center for Mathematics and Its Applications*, volume 32. The Australian National University, 1994.

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

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