Use SeDuMi to Solve LP, SDP and SCOP Problems: Remarks and Examples*

* This file was prepared by Wu-Sheng Lu, Dept. of Electrical and Computer Engineering, University of Victoria, and it was revised on December 2, 2009.

The name of the toolbox, *SeDuMi*, stands for *self-dual minimization* as it implements a self-dual embedding technique for optimization over self-dual homogeneous cones. Below we give comments on the usage of this software for the solution of LP, SDP, and SOCP problems. Numerical examples are included for illustration purposes. Some of the examples are from the book:
A. Antoniou and W.-S. Lu, *Practical Optimization: Algorithms and Engineering Applications*, Springer, 2007.

0. Presently the official web site for public-domain software SeDuMi is http://sedumi.mcmaster.ca/
As of March 2009, the latest version is SeDuMi version 1.1R3. However, this version clashes with MATLAB R2006b. So for users of MATLAB R2006b, the best version of SeDuMi right now appears to be SeDuMi 1.1R2 which can also be downloaded from the above site.

I. LP Problems

The solution of the primal standard-form LP problem

$$\begin{aligned} &\text{minimize} && \boldsymbol{c}^T\boldsymbol{x} && \text{(1a)}\\ &\text{subject to:} && \boldsymbol{Ax}=\boldsymbol{b} && \text{(1b)}\\ & && \boldsymbol{x}\ge\boldsymbol{0} && \text{(1c)}\end{aligned}$$

can be obtained by using command x = sedumi(A,b,c); Alternatively, both the solution of the problem in (1) and the solution of the dual problem

$$\begin{aligned} &\text{maximize} && \boldsymbol{b}^T\boldsymbol{y} && \text{(2a)}\\ &\text{subject to:} && -\boldsymbol{A}^T\boldsymbol{y}\ge-\boldsymbol{c} && \text{(2b)}\end{aligned}$$

can be found by using command [x, y, info] = sedumi(A,b,c); where “info” contains information about validity of the solutions obtained:

(1) pinf = dinf = 0: $\boldsymbol{x}$ is an optimal solution and $\boldsymbol{y}$ certifies optimality, viz. $\boldsymbol{b}^T\boldsymbol{y} = \boldsymbol{c}^T\boldsymbol{x}$ and $\boldsymbol{c} - \boldsymbol{A}^T\boldsymbol{y} \ge 0$.
Stated otherwise, $\boldsymbol{y}$ is an optimal solution to maximize $\boldsymbol{b}^T\boldsymbol{y}$ such that $\boldsymbol{c} - \boldsymbol{A}^T\boldsymbol{y} \ge 0$.

If size(A,2) = length(b), then $\boldsymbol{y}$ solves the linear program maximize $\boldsymbol{b}^T\boldsymbol{y}$ such that $\boldsymbol{c} - \boldsymbol{A}^T\boldsymbol{y} \ge 0$.

(2) pinf = 1: there cannot be $\boldsymbol{x} \ge \boldsymbol{0}$ with $\boldsymbol{Ax} = \boldsymbol{b}$, and this is certified by $\boldsymbol{y}$, viz. $\boldsymbol{b}^T\boldsymbol{y} > 0$ and $\boldsymbol{A}^T\boldsymbol{y} \le 0$.
Thus $\boldsymbol{y}$ is a Farkas solution.

(3) dinf = 1: there cannot be $\boldsymbol{y}$ such that $\boldsymbol{c} - \boldsymbol{A}^T\boldsymbol{y} \ge 0$, and this is certified by $\boldsymbol{x}$, viz. $\boldsymbol{c}^T\boldsymbol{x} < 0$, $\boldsymbol{Ax} = \boldsymbol{b}$, $\boldsymbol{x} \ge \boldsymbol{0}$. Thus $\boldsymbol{x}$ is a Farkas solution.

Example 1 Consider the standard-form LP problem in Example 11.9 [Antoniou and Lu]:

$$\begin{aligned} &\text{minimize} \quad f(x) = 2x_1 + 9x_2 + 3x_3 \\ &\text{subject to:} \quad -2x_1 + 2x_2 + x_3 - x_4 = 1 \\ &\qquad\qquad x_1 + 4x_2 - x_3 - x_5 = 1 \\ &x_1 \ge 0, x_2 \ge 0, x_3 \ge 0, x_4 \ge 0, x_5 \ge 0 \end{aligned} \tag{3}$$

The MATLAB code listed below uses SeDuMi to solve the problem:

```
A = [-2 2 1 -1 0; 1 4 -1 0 -1];
b = [1 1]';
c = [2 9 3 0 0]';
x = sedumi(A,b,c);
```

which gives x = [0 0.3333 0.3333 0 0]'.

<u>Example</u> <u>2</u> Now let us consider the LP problem in Example 11.2 [Antoniou and Lu]:

$$\begin{aligned} &\text{minimize} \quad -x_1 - 4x_2 \\ &\text{subject to:} \quad x_1 \ge 0 \\ &\qquad -x_1 \ge -2 \\ &\qquad x_2 \ge 0 \\ &-x_1 - x_2 + 3.5 \ge 0 \\ &-x_1 - 2x_2 + 6 \ge 0 \end{aligned} \tag{4}$$

By changing the variable ***x*** to ***y*** and defining

$$\boldsymbol{A} = \begin{bmatrix} -1 & 1 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 & 2 \end{bmatrix}, \ \boldsymbol{b} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, \ \boldsymbol{c} = \begin{bmatrix} 0 & 2 & 0 & 3.5 & 6 \end{bmatrix}^T$$

the problem in (4) can be expressed as the one in (2). Hence the LP problem in (4) can be solved by using the following MATLAB code:

```
b = [1 4]';
A = [-1 1 0 1 1; 0 0 -1 1 2];
c = [0 2 0 3.5 6]';
[x, y, info] = sedumi(A,b,c);
```

where y can be taken as the solution of the problem in (4). The result is given by y = [0 3]'. In addition, output "info" provides the following information:

```
     cpusec: 0.0938
       iter: 4
  feasratio: 1
       pinf: 0
       dinf: 0
     numerr: 0
```

II. SOCP Problems

We now consider the SOCP formulation given by Eq. (14.104) (see [Antoniou and Lu]), i.e.,

$$\begin{array}{lll} \text{minimize} & \boldsymbol{b}^T \boldsymbol{x} & \text{(5a)} \\ \text{subject to:} & \left\| \boldsymbol{A}_i^T \boldsymbol{x} + \boldsymbol{c}_i \right\| \le \boldsymbol{b}_i^T \boldsymbol{x} + d_i \ \text{ for } i = 1, \ldots, q & \text{(5b)} \end{array}$$

where $\boldsymbol{b} \in R^{m\times 1}, \boldsymbol{x} \in R^{m\times 1}, \boldsymbol{A}_i \in R^{m\times(n_i-1)}, \boldsymbol{b}_i \in R^{m\times 1}, \boldsymbol{c}_i \in R^{(n_i-1)\times 1}$, and $d_i \in R$ for $1 \le i \le q$. In order to use the toolbox to solve the problem in (5), we define matrix $\boldsymbol{A}_t$, vectors $\boldsymbol{b}_t$ and $\boldsymbol{c}_t$ as follows:

$$\begin{aligned} \boldsymbol{A}_t &= \begin{bmatrix} \boldsymbol{A}_t^{(1)} & \boldsymbol{A}_t^{(2)} & \cdots & \boldsymbol{A}_t^{(q)} \end{bmatrix} \\ \boldsymbol{A}_t^{(i)} &= -\begin{bmatrix} \boldsymbol{b}_i & \boldsymbol{A}_i \end{bmatrix} \\ \boldsymbol{b}_t &= -\boldsymbol{b} \\ \boldsymbol{c}_t &= \begin{bmatrix} \boldsymbol{c}_t^{(1)}; & \boldsymbol{c}_t^{(2)}; & \cdots & \boldsymbol{c}_t^{(q)} \end{bmatrix} \\ \boldsymbol{c}_t^{(i)} &= \begin{bmatrix} d_i; & \boldsymbol{c}_i \end{bmatrix} \end{aligned}$$

where $\boldsymbol{A}_t \in R^{m\times n}, \boldsymbol{b}_t \in R^{m\times 1}$, and $\boldsymbol{c}_t \in R^{n\times 1}$ with $n = \sum_{i=1}^{q} n_i$.

In addition, define a q-dimensional vector $\boldsymbol{q} = [n_1 \ \ n_2 \ \ \ldots \ n_q]$ which describes the dimensions of the q conic constraints involved in (5b).

Once the data set $\{\boldsymbol{A}_t, \boldsymbol{b}_\text{t}, \boldsymbol{c}_t\}$ and vector $\boldsymbol{q}$ have been prepared, the MATLAB commands for solving the SOCP problem in (5) are as follows:

```
K.q = q;
[xs,ys,info] = sedumi(At,bt,ct,K);
info
x = ys;
```

Example 3 Consider the problem described in Example 14.5 in [Antoniou and Lu]: Find the shortest distance between the two ellipses which are defined by

$$c_1(\boldsymbol{x}) = -\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} \frac{1}{4} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ 0 \end{bmatrix} + \frac{3}{4} \ge 0$$

$$c_2(\boldsymbol{x}) = -\frac{1}{8}\begin{bmatrix} x_3 & x_4 \end{bmatrix} \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} x_3 & x_4 \end{bmatrix} \begin{bmatrix} \frac{11}{2} \\ \frac{13}{2} \end{bmatrix} - \frac{35}{2} \ge 0$$

The problem can be formulated as the constrained minimization problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) = (x_1 - x_3)^2 + (x_2 - x_4)^2 \\ &\text{subject to:} \quad c_1(\boldsymbol{x}) \ge 0 \ \text{ and } \ c_2(\boldsymbol{x}) \ge 0 \end{aligned}$$

By introducing an upper bound δ for the square root of the objective function, the above problem can be expressed as

$$\begin{aligned} &\text{minimize} \quad \delta \\ &\text{subject to:} \quad \left[(x_1 - x_3)^2 + (x_2 - x_4)^2\right]^{1/2} \le \delta \\ &\left[x_1 \quad x_2\right]\begin{bmatrix} 1/4 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \left[x_1 \quad x_2\right]\begin{bmatrix} 1/2 \\ 0 \end{bmatrix} \le \frac{3}{4} \\ &\left[x_3 \quad x_4\right]\begin{bmatrix} 5/8 & 3/8 \\ 3/8 & 5/8 \end{bmatrix}\begin{bmatrix} x_3 \\ x_4 \end{bmatrix} - \left[x_3 \quad x_4\right]\begin{bmatrix} 11/2 \\ 13/2 \end{bmatrix} \le -\frac{35}{2} \end{aligned}$$

If we augment the decision vector by including the upper bound δ in it, i.e., $\boldsymbol{x} = [\,\delta \ x_1\ x_2\ x_3\ x_4]^T$, then the above problem is equivalent to the problem in (5) with q = 3,

$$\boldsymbol{b} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

$$\boldsymbol{A}_1^T = \begin{bmatrix} 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 \end{bmatrix}, \ \boldsymbol{A}_2^T = \begin{bmatrix} 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{A}_3^T = \begin{bmatrix} 0 & 0 & 0 & -0.7071 & -0.7071 \\ 0 & 0 & 0 & -0.3536 & 0.3536 \end{bmatrix}$$

$$\boldsymbol{b}_1 = \boldsymbol{b}, \ \boldsymbol{b}_2 = \boldsymbol{0}, \ \boldsymbol{b}_3 = \boldsymbol{0}$$

$$\boldsymbol{c}_1 = \boldsymbol{0}, \ \boldsymbol{c}_2 = \begin{bmatrix} -0.5 & 0 \end{bmatrix}^T, \ \boldsymbol{c}_3 = \begin{bmatrix} 4.2426 & -0.7071 \end{bmatrix}^T$$

$$d_1 = 0, \ d_2 = 1, \ \text{and} \ d_3 = 1.$$

The MATLAB code listed below solves the SOCP problem using SeDuMi.

```
b = [1 0 0 0 0]';
A1 = [0 -1 0 1 0; 0 0 1 0 -1]';
A2 = [0 0.5 0 0 0; 0 0 1 0 0]';
A3 = [0 0 0 -0.7071 -0.7071; 0 0 0 -0.3536 0.3536]';
b1 = b;
b2 = zeros(5,1);
b3 = b2;
c1 = [0 0]';
c2 = [-0.5 0]';
c3 = [4.2426 -0.7071]';
d1 = 0;
d2 = 1;
d3 = 1;
At1 = -[b1 A1];
At2 = -[b2 A2];
At3 = -[b3 A3];
```

```
At = [At1 At2 At3];
bt = -b;
ct1 = [d1; c1];
ct2 = [d2; c2];
ct3 = [d3; c3];
ct = [ct1; ct2; ct3];
K.q = [size(At1,2)  size(At2,2)  size(At3,2)];
[xs, ys, info] = sedumi(At,bt,ct,K);
x = ys;
r_s = x(2:3);
s_s = x(4:5);
disp(‘solution points in regions R and S are:’)
[r_s s_s]
disp(‘minimum distance:’)
norm(r_s – s_s)
```

It was found that r_s = [2.0447 0.8527]’ and s_s = [2.5448 2.4857]’ which give the minimum distance 1.7078. Additional information provided by the toolbox is as follows:

```
info =
       cpusec: 0.0469
         iter: 9
    feasratio: 1.0000
         pinf: 0
         dinf: 0
       numerr: 0
```

III. SOCP Problems with Linear Constraints

SeDuMi can be used to solve SOCP problems with additional linear constraints. As an example, we consider the following SOCP problem

$$\text{minimize} \quad \boldsymbol{b}^T\boldsymbol{x} \tag{6a}$$

$$\text{subject to:} \quad \boldsymbol{D}^T\boldsymbol{x} + \boldsymbol{f} \geq \boldsymbol{0} \tag{6b}$$

$$\left\| \boldsymbol{A}_i^T\boldsymbol{x} + \boldsymbol{c}_i \right\| \leq \boldsymbol{b}_i^T\boldsymbol{x} + d_i \quad \text{for } i = 1, \ldots, q \tag{6c}$$

where $\boldsymbol{D} \in R^{m\times p}$ and $\boldsymbol{f} \in R^{p\times 1}$ and other entries are defined in part II. In order to use the toolbox to solve the problem in (6), we define matrix $\boldsymbol{A}_t$, vectors $\boldsymbol{b}_t$ and $\boldsymbol{c}_t$, which are obtained by augmenting the corresponding quantities used for the problem in (5), as follows:

$$\begin{aligned}
\boldsymbol{A}_t &= \begin{bmatrix} -\boldsymbol{D} & \boldsymbol{A}_t^{(1)} & \boldsymbol{A}_t^{(2)} & \cdots & \boldsymbol{A}_t^{(q)} \end{bmatrix} \\
\boldsymbol{A}_t^{(i)} &= -\begin{bmatrix} \boldsymbol{b}_i & \boldsymbol{A}_i \end{bmatrix} \\
\boldsymbol{b}_t &= -\boldsymbol{b} \\
\boldsymbol{c}_t &= \begin{bmatrix} \boldsymbol{f}; & \boldsymbol{c}_t^{(1)}; & \boldsymbol{c}_t^{(2)}; & \cdots & \boldsymbol{c}_t^{(q)} \end{bmatrix} \\
\boldsymbol{c}_t^{(i)} &= \begin{bmatrix} d_i; & \boldsymbol{c}_i \end{bmatrix}
\end{aligned}$$

Now define K.l = p where p is the number of linear constraints in (6b). And as before, define a q-dimensional vector $\boldsymbol{q} = [n_1 \;\; n_2 \;\; \ldots \; n_q]$ to describe the dimensions of the q conic constraints in (6c). The MATLAB commands to solve the SOCP problem in (5) are as follows:

```
K.l = p;
K.q = q;
[xs,ys,info] = sedumi(At,bt,ct,K);
info
x = ys;
```

Example 4 Consider a problem similar to the one described in Example 14.5 of [Antoniou and Lu], to that we now add three additional linear constraints for variables x_3 and x_4 as follows:

$$\begin{aligned} &-x_3 + 2.4 \geq 0 \\ &x_4 - 2.4 \geq 0 \\ &1.5x_3 - x_4 + 2.4 \geq 0 \end{aligned}$$

It can be readily verified that the problem at hand is a minimum distance problem for the ellipse $\mathcal{R}$ and convex body $\mathcal{S}$ shown in the next figure.

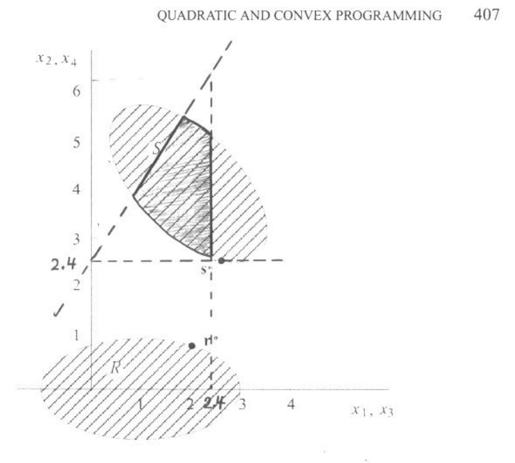

Figure 13.5. Distance between two ellipses (Example 13.5).

The MATLAB code listed below solves this SOCP problem using SeDuMi.

```
D = [0 0 0; 0 0 0; 0 0 0; -1 0 1.5; 0 1 -1];
f = [2.4 -2.4 2.4]’;
b = [1 0 0 0 0]’;
A1 = [0 -1 0 1 0; 0 0 1 0 -1]’;
A2 = [0 0.5 0 0 0; 0 0 1 0 0]’;
A3 = [0 0 0 -0.7071 -0.7071; 0 0 0 -0.3536 0.3536]’;
b1 = b;
b2 = zeros(5,1);
b3 = b2;
c1 = [0 0]’;
c2 = [-0.5 0]’;
c3 = [4.2426 -0.7071]’;
d1 = 0;
```

```
d2 = 1;
d3 = 1;
At1 = -[b1 A1];
At2 = -[b2 A2];
At3 = -[b3 A3];
At = [-D At1 At2 At3];
bt = -b;
ct1 = [d1; c1];
ct2 = [d2; c2];
ct3 = [d3; c3];
ct = [f; ct1; ct2; ct3];
K.l = size(D,2);
K.q = [size(At1,2)  size(At2,2)  size(At3,2)];
[xs, ys, info] = sedumi(At,bt,ct,K);
x = ys;
r_s = x(2:3);
s_s = x(4:5);
disp('solution points in regions R and S are:')
[r_s s_s]
disp('minimum distance:')
norm(r_s – s_s)
```

It was found that r_s = [1.9518 0.8795]' and s_s = [2.4000 2.5362]' which give the minimum distance 1.7163. Additional information provided by the toolbox are as follows:

```
info =
      cpusec: 0.6406
        iter: 13
   feasratio: 1.0000
        pinf: 0
        dinf: 0
      numerr: 0
```

IV. SDP Problems

Consider the "standard" dual SDP problem

$$\text{minimize} \quad \boldsymbol{c}^T \boldsymbol{y} \tag{7a}$$
$$\text{subject to: } \boldsymbol{F}_0 + \boldsymbol{y}_1\boldsymbol{F}_1 + \cdots \boldsymbol{y}_p\boldsymbol{F}_p \succeq \boldsymbol{0} \tag{7b}$$

where vector $\boldsymbol{y} = [y_1 \ \ y_2 \ \ \ldots \ \ y_p]^T$, and $\boldsymbol{F}_i$ are symmetric matrices. In order to use SeDuMi to solve the SDF problem, we define

$\boldsymbol{b}_t = -\boldsymbol{c}$;
$\boldsymbol{c}_t = \text{vec}(\boldsymbol{F}_0)$;
$\boldsymbol{A}_t(:, i) = -\text{vec}(\boldsymbol{F}_i)$ for $i = 1, 2, \ldots, p$

where vec$(\boldsymbol{F})$ converts matrix $\boldsymbol{F}$ into a column vector by stacking all columns of $\boldsymbol{F}$. The MATLAB code using commands from SeDuMi is as follows.

```
p = length(c);
bt = -c;
ct = vec(F0);
for i = 1:p, At(:,i) = -vec(Fi); end
K.s = size(F0,1);
[x, y, info] = sedumi(At,bt,ct,K);
info
```

where y gives the solution of the problem in (7) and info provides information about the validity of the solution.

Example 5 Solve the problem in Example 14.1 of [Antoniou and Lu]: Find scalars y_1, y_2, and y_3 such that the maximum eigenvalue of $\boldsymbol{F} = \boldsymbol{A}_0 + y_1\boldsymbol{A}_1 + y_2\boldsymbol{A}_2 + y_3\boldsymbol{A}_3$ with

$$\boldsymbol{A}_0 = \begin{bmatrix} 2 & -0.5 & -0.6 \\ -0.5 & 2 & 0.4 \\ -0.6 & 0.4 & 3 \end{bmatrix},\ \boldsymbol{A}_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \boldsymbol{A}_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \boldsymbol{A}_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

is minimized. This problem can be formulated as the SDP problem

$$\text{minimize} \quad t \qquad \text{(8a)}$$
$$\text{subject to:} \quad t\boldsymbol{I} - (\boldsymbol{A}_0 + y_1\boldsymbol{A}_1 + y_2\boldsymbol{A}_2 + y_3\boldsymbol{A}_3) \succeq \boldsymbol{0} \qquad \text{(8b)}$$

With $\boldsymbol{y} = [y_1\ \ y_2\ \ y_3\ \ t]^T$, $F_0 = -\boldsymbol{A}_0$, $F_1 = -\boldsymbol{A}_1$, $F_2 = -\boldsymbol{A}_2$, and $\boldsymbol{c} = [0\ \ 0\ \ 0\ \ 1]^T$, problem (8) becomes the standard SDP problem in (7). The MATLAB code listed below uses SeDuMi to solve the problem in (8).

```
A0 = [2 -0.5 -0.6; -0.5 2 0.4; -0.6 0.4 3];
A1 = [0 1 0; 1 0 0; 0 0 0];
A2 = [0 0 1; 0 0 0; 1 0 0];
A3 = [0 0 0; 0 0 1; 0 1 0];
F0 = -A0;
F1 = -A1;
F2 = -A2;
F3 = -A3;
F4 = eye(3);
At = -[vec(F1) vec(F2) vec(F3) vec(F4)];
bt = -[0 0 0 1]';
ct = vec(F0);
K.s = size(F0,1);
[x,y,info] = sedumi(At,bt,ct,K);
info
```

The result obtained is given by y = [0.5 0.6 -0.4 3]'; which says the minimum of the largest eigenvalue is 3 that can be achieved by the choice $y_1 = 0.5$, $y_2 = 0.6$, and $y_3 = -0.4$. The 'info' gives the following:

```
cpusec: 0.5000
iter: 5
feasratio: 1.0000
pinf: 0
dinf: 0
numerr: 0
```

V. SDP Problems with Linear Constraints

In principal linear constraints can be formulated as SDP constraints (see Chap. 14 of [Antoniou and Lu]), but one can take the advantage of SeDuMi that allows both linear and SDP constraints simultaneously to make the MATLAB code more efficient. To this end, we consider the problem

$$\text{minimize} \quad \boldsymbol{c}^T \boldsymbol{y} \qquad (9a)$$
$$\text{subject to:} \quad \boldsymbol{A}\boldsymbol{y} \geq \boldsymbol{b} \qquad (9b)$$
$$\boldsymbol{F}_0 + y_1\boldsymbol{F}_1 + \cdots y_p\boldsymbol{F}_p \succeq \boldsymbol{0} \qquad (9c)$$

Similar to that in part (IV), we define

$\boldsymbol{b}_t = -\boldsymbol{c}$;
$\boldsymbol{c}_t = \text{vec}(\boldsymbol{F}_0)$;
$\boldsymbol{A}_t(:, i) = -\text{vec}(\boldsymbol{F}_i)$ for $i = 1, 2, \ldots, p$

The MATLAB code using commands from SeDuMi is as follows.

```
p = length(c);
btt = -c;
ctt = [-b; vec(F0)];
for i = 1:p, At(:,i) = -vec(Fi); end
Att  = [-A; At];
K.l = size(A,1);
K.s = size(F0,1);
[x, y, info] = sedumi(At,bt,ct,K);
info
```

where y gives the solution of the problem in (9) and info provides information about the validity of the solution.

__Example 6__ We now consider a problem similar to the one in Example 5, but we add additional constraints that parameters y_1, y_2, y_3 satisfy $0.7 \leq y_1 \leq 1$, $0 \leq y_2 \leq 0.3$, and $y_3 \geq 0$. These additional linear constraints can be put into a matrix form $\boldsymbol{A}\boldsymbol{y} \geq \boldsymbol{b}$ with

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \ \boldsymbol{b} = \begin{bmatrix} 0.7 \\ -1 \\ 0 \\ -0.3 \\ 0 \end{bmatrix}$$

where vector y is defined as in Example 5, i.e., $\boldsymbol{y} = [y_1 \ \ y_2 \ \ y_3 \ \ t]^T$. The MATLAB code solving this problem using SeDuMi is as follows:

```
A = [1 0 0 0; -1 0 0 0; 0 1 0 0; 0 -1 0 0; 0 0 1 0];
b = [0.7 -1 0 -0.3 0]';
A0 = [2 -0.5 -0.6; -0.5 2 0.4; -0.6 0.4 3];
```

```
A1 = [0 1 0; 1 0 0; 0 0 0];
A2 = [0 0 1; 0 0 0; 1 0 0];
A3 = [0 0 0; 0 0 1; 0 1 0];
F0 = -A0;
F1 = -A1;
F2 = -A2;
F3 = -A3;
F4 = eye(3);
At = -[vec(F1) vec(F2) vec(F3) vec(F4)];
Att = [-A; At];
btt = -[0 0 0 1]';
ct = vec(F0);
ctt = [-b; ct];
K.l = size(A,1);
K.s = size(F0,1);
[x,y,info] = sedumi(Att,btt,ctt,K);
info
```

The result obtained is given by y = [1.0 0.3 -0.0 3.1556]'; which says the minimum of the largest eigenvalue is 3.1556 that can be achieved by the choice $y_1 = 1.0$, $y_2 = 0.3$, and $y_3 = 0$. The info gives the following:

```
iter: 9
feasratio: 0.9967
pinf: 0
dinf: 0
numerr: 0
timing: [0.0313 0.2969 0.0156]
cpusec: 0.3438
```

VI. Other Problems

6.1 [x, y, info] = sedumi(A, b, 0) solves the feasibility problem: Find $\boldsymbol{x}$ such that $\boldsymbol{Ax} = \boldsymbol{b}$ and $\boldsymbol{x} \geq \boldsymbol{0}$.

Example 7 With $\boldsymbol{A}$ and $\boldsymbol{b}$ given in Example 1, [x, y, info] = sedumi(A, b, 0) yields $\boldsymbol{x} = [1.3130\ \ 1.2353\ \ 2.8405\ \ 1.6851\ \ 2.4136]^T$ and

```
info =
cpusec: 0
iter: 1
feasratio: 1
pinf: 0
dinf: 0
numerr: 0
```

6.2 [x, y, info] = sedumi(A, 0, c) solves the feasibility problem: Find $\boldsymbol{y}$ such that $\boldsymbol{A}^T\boldsymbol{y} \leq \boldsymbol{c}$.

Example 8 Let

$$\boldsymbol{A} = \begin{bmatrix} -1 & 1 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 & 2 \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} 0 & 2 & 0 & 3.5 & 6 \end{bmatrix}^T$$

Applying `[x, y, info] = sedumi(A, 0, c)` yields

$\boldsymbol{y} = [0.9810 \;\; 1.2670]^T$ and

```
info =
cpusec: 0.0625
iter: 3
feasratio: 1
pinf: 0
dinf: 0
numerr: 0
```