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

*The notes were prepared by Wu-Sheng Lu, Dept. of Electrical and Computer Engineering, University of Victoria; the latest revision was done on March 13, 2012.

SeDuMi was originally written by Jos F. Sturm in 1999. The name of the toolbox stands for self-dual minimization as it implements a self-dual embedding technique for optimization over self-dual homogeneous cones. Presently the official web site for the software’s maintenance and development is http://sedumi.ie.lehigh.edu/ at Lehigh University. 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 A. Antoniou and W.-S. Lu, Practical Optimization: Algorithms and Engineering Applications, Springer, 2007.

0. As of March 2012, the latest version is SeDuMi version 1.3 officially released in April 2010. Although the notes given below were based on SeDuMi 1.1R2, it is believed that the example codes given here also work with version 1.3.

I. LP Problems

The solution of the primal standard-form LP problem

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to:} & \quad Ax = b \\
& \quad x \geq 0
\end{align*}
\]  

(1a)\hspace{1cm} (1b)\hspace{1cm} (1c)

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

\[
\begin{align*}
\text{maximize} & \quad b^T y \\
\text{subject to:} & \quad -A^T y \geq -c
\end{align*}
\]  

\( (2a) \hspace{1cm} (2b) \)

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

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

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

(2) \( \text{pinf} = 1: \) there cannot be \( x \geq 0 \) with \( Ax = b, \) and this is certified by \( y, \) viz. \( b^T y > 0 \) and \( A^T y \leq 0. \)

Thus \( y \) is a Farkas solution.

(3) \( \text{dinf} = 1: \) there cannot be \( y \) such that \( c - A^T y \geq 0, \) and this is certified by \( x, \) viz. \( c^T x < 0, Ax = b, x \geq 0. \) Thus \( x \) is a Farkas solution.
Example 1 Consider the standard-form LP problem in Example 11.9 [Antoniou and Lu]:

\[
\begin{align*}
\text{minimize} & \quad f(x) = 2x_1 + 9x_2 + 3x_3 \\
\text{subject to:} & \quad -2x_1 + 2x_2 + x_3 - x_4 = 1 \\
& \quad x_1 + 4x_2 - x_3 - x_4 = 1 \\
& \quad x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0
\end{align*}
\] (3)

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

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

which gives \(x = [0 \ 0.3333 \ 0.3333 \ 0 \ 0]'\).

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

\[
\begin{align*}
\text{minimize} & \quad -x_1 - 4x_2 \\
\text{subject to:} & \quad x_1 \geq 0 \\
& \quad -x_1 \geq -2 \\
& \quad x_2 \geq 0 \\
& \quad -x_1 - x_2 + 3.5 \geq 0 \\
& \quad -x_1 - 2x_2 + 6 \geq 0
\end{align*}
\] (4)

By changing the variable \(x\) to \(y\) and defining

\[
A = \begin{bmatrix} -1 & 1 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, \quad c = [0 \ 2 \ 0 \ 3.5 \ 6]'\]
\]

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:

```matlab
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.,

minimize $\mathbf{b}^T \mathbf{x}$  \hspace{1cm} (5a)

subject to: $\|A_i^T \mathbf{x} + c_i\| \leq b_i^T \mathbf{x} + d_i$ for $i = 1, \ldots, q$  \hspace{1cm} (5b)

where $\mathbf{b} \in \mathbb{R}^m \times 1$, $\mathbf{x} \in \mathbb{R}^n$, $A_i \in \mathbb{R}^{m \times (n_i-1)}$, $b_i \in \mathbb{R}^m$, $c_i \in \mathbb{R}^{(n_i-1) \times 1}$, and $d_i \in \mathbb{R}$ for $1 \leq i \leq q$. In order to use the toolbox to solve the problem in (5), we define matrix $A_t$, vectors $b_t$ and $c_t$ as follows:

$$A_t = \begin{bmatrix} A_1^{(1)} & A_2^{(1)} & \cdots & A_q^{(1)} \\ A_1^{(2)} & \cdots & A_q^{(2)} \\ \vdots & \ddots & \vdots \\ A_1^{(q)} & \cdots & A_q^{(q)} \end{bmatrix}$$

$$b_t = -b$$

$$c_t = \begin{bmatrix} c_1^{(1)} & c_2^{(1)} & \cdots & c_q^{(1)} \\ c_1^{(2)} & \cdots & c_q^{(2)} \\ \vdots & \ddots & \vdots \\ c_1^{(q)} & \cdots & c_q^{(q)} \end{bmatrix}$$

$$c_t = \begin{bmatrix} d_1 & \cdots & d_q \end{bmatrix}$$

where $A_i \in \mathbb{R}^{m \times n_i}$, $b_i \in \mathbb{R}^m$, and $c_i \in \mathbb{R}^{n_i}$ with $n = \sum_{i=1}^q n_i$.

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

Once the data set $\{A_t, b_t, c_t\}$ and vector $\mathbf{q}$ 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

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

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

The problem can be formulated as the constrained minimization problem
minimize \( f(x) = (x_1 - x_2)^2 + (x_2 - x_3)^2 \)
subject to: \( c_1(x) \geq 0 \) and \( c_2(x) \geq 0 \)

By introducing an upper bound \( \delta \) for the square root of the objective function, the above problem can be expressed as

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

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

\[
b = [1 \ 0 \ 0 \ 0 \ 0]^T
\]
\[
A_1^T = [0 \ -1 \ 0 \ 1 \ 0] \quad A_2^T = [0 \ 0.5 \ 0 \ 0 \ 0]
\]
\[
A_3^T = [0 \ 0 \ 0 \ -0.7071 \ -0.7071] \quad [0 \ 0 \ 0 \ -0.3536 \ 0.3536]
\]
\[
b_1 = b, \ b_2 = 0, \ b_3 = 0
\]
\[
c_1 = 0, \ c_2 = [-0.5 \ 0]^T, \ c_3 = [4.2426 \ -0.7071]^T
\]
\[
d_1 = 0, \ d_2 = 1, \ \text{and} \ d_3 = 1.
\]

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

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:

\[ \text{info} = \]
\begin{align*}
\text{cpusec} & : 0.0469 \\
\text{iter} & : 9 \\
\text{feasratio} & : 1.0000 \\
\text{pinf} & : 0 \\
\text{dinf} & : 0 \\
\text{numerr} & : 0
\end{align*}

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

\begin{align*}
\text{minimize} & \quad b^T x \quad \text{(6a)} \\
\text{subject to:} & \quad D^T x + f \geq 0 \quad \text{(6b)} \\
& \quad \|A_i^T x + c_i\| \leq b_i^T x + d_i \quad \text{for } i = 1, \ldots, q \quad \text{(6c)}
\end{align*}

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

\( A_i = \begin{bmatrix} -D & A_i^{(1)} & A_i^{(2)} & \cdots & A_i^{(q)} \end{bmatrix} \)
\( A_i^{(i)} = [-b_i \ \ A_i] \)
\( b_i = -b \)
\( c_i = [f; \ c_i^{(1)}; \ c_i^{(2)}; \ \cdots \ c_i^{(q)}] \)
\( c_i^{(i)} = [d_i; \ c_i] \)
Now define $K.l = p$ where $p$ is the number of linear constraints in (6b). And as before, define a $q$-dimensional vector $q = [n_1 \ n_2 \ \ldots \ \ n_q]$ to describe the dimensions of the $q$ conic constraints in (6c). The MATLAB commands to solve the SOCP problem in (5) are as follows:

```matlab
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:

\[-x_3 + 2.4 \geq 0
\]

\[x_4 - 2.4 \geq 0
\]

\[1.5x_3 - x_4 + 2.4 \geq 0
\]

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

![Figure 13.5. Distance between two ellipses (Example 13.5)](image)

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

```matlab
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;
```
\(d_2 = 1;\)
\(d_3 = 1;\)
\(\mathbf{A}_1 = [-b_1 \mathbf{A}_1];\)
\(\mathbf{A}_2 = [-b_2 \mathbf{A}_2];\)
\(\mathbf{A}_3 = [-b_3 \mathbf{A}_3];\)
\(\mathbf{A} = [-D \mathbf{A}_1 \mathbf{A}_2 \mathbf{A}_3];\)
\(b_t = -b;\)
\(c_{t1} = [d_1; c_1];\)
\(c_{t2} = [d_2; c_2];\)
\(c_{t3} = [d_3; c_3];\)
\(c_t = [f; c_{t1}; c_{t2}; c_{t3}];\)
\(K.l = \text{size}(D,2);\)
\(K.q = \text{size}(\mathbf{A}_1,2) \times \text{size}(\mathbf{A}_2,2) \times \text{size}(\mathbf{A}_3,2);\)
\([xs, ys, info] = \text{sedumi}(\mathbf{A}, b_t, c_t, K);\)
\(x = ys;\)
\(r_s = x(2:3);\)
\(s_s = x(4:5);\)
\(\text{disp('solution points in regions R and S are:')}\)
\([r_s s_s]\)
\(\text{disp('minimum distance:')}\)
\(\text{norm}(r_s - s_s)\)

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:

\(\text{info =}\)
\(\text{cpusec: 0.6406}\)
\(\text{iter: 13}\)
\(\text{feasratio: 1.0000}\)
\(\text{p inf: 0}\)
\(\text{d inf: 0}\)
\(\text{numerr: 0}\)

IV. SDP Problems

Consider the “standard” dual SDP problem

\[
\begin{align*}
\text{minimize} & \quad c^T y & \quad (7a) \\
\text{subject to:} & \quad F_0 + y_1 F_1 + \cdots + y_p F_p \succeq 0 & \quad (7b)
\end{align*}
\]

where vector \(y = [y_1\; y_2\; \ldots\; y_p]^T\), and \(F_i\) are symmetric matrices. In order to use SeDuMi to solve the SDF problem, we define

\(b_t = -c;\)
\(c_t = \text{vec}(F_0);\)
\(A_i(:, i) = -\text{vec}(F_i) \quad \text{for} \; i = 1, 2, \ldots, p\)

where \(\text{vec}(F)\) converts matrix \(F\) into a column vector by stacking all columns of \(F\). The MATLAB code using commands from SeDuMi is as follows.
\( p = \text{length}(c); \)
\( bt = -c; \)
\( ct = \text{vec}(F0); \)
for \( i = 1:p, \ At(:,i) = -\text{vec}(Fi); \) end
\( K.s = \text{size}(F0,1); \)
\([x, y, \text{info}] = \text{sedumi}(At,bt,ct,K); \)
\text{info}

where \( y \) gives the solution of the problem in (7) and \( \text{info} \) 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 \( F = A_0 + y_1A_1 + y_2A_2 + y_3A_3 \) with

\[
A_0 = \begin{bmatrix} 2 & -0.5 & -0.6 \\ -0.5 & 2 & 0.4 \\ -0.6 & 0.4 & 3 \end{bmatrix}, \ A_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \ A_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \ A_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}
\]

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

\[
\begin{align*}
\text{minimize} & \quad t \\
\text{subject to:} & \quad tI - (A_0 + y_1A_1 + y_2A_2 + y_3A_3) \succeq 0
\end{align*}
\]  

(8a) (8b)

With \( y = [y_1 \ y_2 \ y_3 \ t]^T, F_0 = -A_0, F_1 = -A_1, F_2 = -A_2, \) and \( c = [0 \ 0 \ 0 \ 1]^T, \) problem (8) becomes the standard SDP problem in (7). The MATLAB code listed below uses SeDuMi to solve the problem in (8).

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

The result obtained is given by \( y = [0.5 \ 0.6 \ -0.4 \ 3]^T; \) 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:

\text{cpusec:} 0.5000 \\
\text{iter:} 5 \\
\text{feasratio:} 1.0000 \\
\text{pinf:} 0 \\
\text{dinf:} 0 \\
\text{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

\[
\begin{align*}
\text{minimize} & \quad c^T y \\
\text{subject to:} & \quad Ay \geq b \\
& \quad F_0 + y_1 F_1 + \cdots + y_p F_p \succeq 0
\end{align*}
\]

Similar to that in part (IV), we define

\[
\begin{align*}
b_t &= -c \\
c_t &= \text{vec}(F_0) \\
A_t(:, i) &= -\text{vec}(F_i) \quad \text{for } i = 1, 2, \ldots, p
\end{align*}
\]

The MATLAB code using commands from SeDuMi is as follows.

\[
p = \text{length}(c); \\
btt = -c; \\
ctt = [-b; \text{vec}(F0)]; \\
\text{for } i = 1:p, \quad \text{At(:,} i) = -\text{vec}(F_i); \quad \text{end} \\
\text{Att} = [-A; \text{At}]; \\
\text{K.l} = \text{size}(A,1); \\
\text{K.s} = \text{size}(F0,1); \\
\text{[x, y, info]} = \text{sedumi(Att,btt,ctt,K)}; \\
\text{info}
\]

where \(y\) gives the solution of the problem in (9) and \(\text{info}\) 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 \(0.7 \leq y_i \leq 1, 0 \leq y_2 \leq 0.3, \text{ and } y_3 \geq 0\). These additional linear constraints can be put into a matrix form \(Ay \geq b\) with

\[
A = \begin{bmatrix}
1 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & -1 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}, \quad b = \begin{bmatrix}
0.7 \\
-1 \\
0 \\
-0.3 \\
0
\end{bmatrix}
\]

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

\[
\begin{align*}
A &= [1 \ 0 \ 0 \ 0; -1 \ 0 \ 0; 0 \ 1 \ 0 \ 0; 0 \ -1 \ 0 \ 0; 0 \ 0 \ 1 \ 0]; \\
b &= [0.7 \ -1 \ 0 \ -0.3 \ 0]^T; \\
A0 &= [2 \ -0.5 \ -0.6; -0.5 \ 2 \ 0.4; -0.6 \ 0.4 \ 3];
\end{align*}
\]
\[
A1 = \begin{bmatrix} 0 & 1 & 0; & 1 & 0 & 0; & 0 & 0 & 0 \end{bmatrix};
A2 = \begin{bmatrix} 0 & 0 & 1; & 0 & 0 & 0; & 1 & 0 & 0 \end{bmatrix};
A3 = \begin{bmatrix} 0 & 0 & 0; & 0 & 0 & 1; & 0 & 1 & 0 \end{bmatrix};
F0 = -A0;
F1 = -A1;
F2 = -A2;
F3 = -A3;
F4 = \text{eye}(3);
At = -[\text{vec}(F1) \text{ vec}(F2) \text{ vec}(F3) \text{ vec}(F4)];
\]

The result obtained is given by \( y = [1.0, 0.3, -0.0, 3.1556]^T \); 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 \( [x, y, info] = \text{sedumi}(A, b, 0) \) solves the feasibility problem: Find \( x \) such that \( Ax = b \) and \( x \geq 0 \).

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

info =

cpusec: 0
iter: 1
feasratio: 1
pinf: 0
dinf: 0
numerr: 0

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

Example 8 Let

\[
A = \begin{bmatrix}
-1 & 1 & 0 & 1 & 1 \\
0 & 0 & -1 & 1 & 2
\end{bmatrix}, \quad c = \begin{bmatrix}
0 & 2 & 0 & 3.5 & 6
\end{bmatrix}^T
\]
Applying \([x, y, \text{info}] = \text{sedumi}(A, 0, c)\) yields

\[y = [0.9810 \quad 1.2670]^T\]

and

\[
\text{info} = \\
cpusec: 0.0625 \\
it: 3 \\
feasratio: 1 \\
pinf: 0 \\
dinf: 0 \\
numerr: 0
\]