# LINEAR MATRIX INEQUALITIES AND MATLAB LMI TOOLBOX

**B. Erkus[1] and Y.J. Lee[2]**

(1/20/2004)

## ABSTRACT

This document gives a brief introduction to linear matrix inequalities (LMIs). First section gives the definition and various representations of the LMIs and common problems involving LMIs. The second section introduces MATLAB LMI Toolbox and explains some of the useful functions by example codes.

## INTRODUCTION TO LMIS

### Definition

The matrix inequality

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}_0 + x_1\mathbf{F}_1 + \ldots + x_n\mathbf{F}_n > 0 \qquad (1)$$

is called a *linear matrix inequality* where $\mathbf{F}(\mathbf{x})$ is an affine function of the real vector $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \ldots & x_n \end{bmatrix}^{\mathrm{T}}$, $\mathbf{F}_0, \mathbf{F}_1, \ldots, \mathbf{F}_n$ are real symmetric matrices, and $\mathbf{x}$ is a vector of decision variables.

### Different Representations

- In most applications, LMIs do not naturally arise in the canonical form (1), but rather in the form

$$\mathbf{L}(\mathbf{X}_1, \ldots, \mathbf{X}_n) < \mathbf{R}(\mathbf{X}_1, \ldots, \mathbf{X}_n) \qquad (2)$$

  where $\mathbf{L}(\cdot)$ and $\mathbf{R}(\cdot)$ are affine functions of some structured matrix variables $\mathbf{X_1}, \ldots, \mathbf{X_n}$. For example,

$$\mathbf{AX} + \mathbf{XA}' < 0 \qquad (3)$$

  is an LMI where $\mathbf{X}$ is the decision variable. LMI (3) can be represented in the form of (1) using some algebra.

---

[1] Grad. Student, Civil Engineering Dept., Univ. of Southern California, erkus@usc.edu.

[2] Grad. Student, Civil Engineering Dept., Univ. of Southern California, yongjeil@usc.edu.

- Some authors prefer to use

$$\mathbf{F}(\mathbf{x}) = x_1\mathbf{F}_1 + \ldots + x_n\mathbf{F}_n > -\mathbf{F}_0 \tag{4}$$

Note that left hand side of (1) is affine and left hand side of (2) is linear in $x$.

**Linearity**: A function $f(x)$ is a linear function of $x$ if $f(\alpha x_1 + \beta x_2) = \alpha f(x_1) + \beta f(x_2)$ for all scalar $\alpha$ and $\beta$.

**Affinity**: A function $g(x)$ is an affine function of $x$ if it can be written as $g(x) = f(x) + a$ where $f(x)$ is a linear function and $a$ is a constant.

- A set of LMIs can always be converted to one LMI, i.e., suppose we have the two LMIs

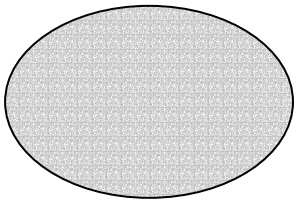
$$\mathbf{F}_0(\mathbf{X}_0) < \mathbf{Q}_0 \quad \text{and} \quad \mathbf{F}_1(\mathbf{X}_1) < \mathbf{Q}_1. \tag{5}$$

This is equivalent to the single LMI $\mathbf{F}(\mathbf{X}) < \mathbf{Q}$ where $\mathbf{F}(\mathbf{X}) = diag\left(\mathbf{F}_0(\mathbf{X}_0), \mathbf{F}_1(\mathbf{X}_1)\right)$ and $\mathbf{Q} = diag\left(\mathbf{Q}_0, \mathbf{Q}_1\right)$.

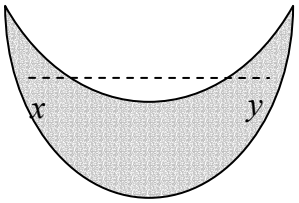
## Properties

- The feasible solution set of (1), $\{x \mid \mathbf{F}(x) > 0\}$ is a convex set. This is an important property since powerful numerical solution techniques are available for the problems involving convex solution sets.

**Convexity**: A set $C \subset \mathbf{R}^n$ is convex if $\mu\mathbf{x}_1 + (1-\mu)\mathbf{x}_2 \in C$ for any $\mathbf{x}_1, \mathbf{x}_2 \in C$ and any $\mu \in [0,1]$.



CONVEX




NOT CONVEX

- LMI (1) is a *strict* inequality. The *nonstrict* version is,

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}_0 + x_1\mathbf{F}_1 + \ldots + x_n\mathbf{F}_n \geq 0. \tag{6}$$

- **The Schur complement formula**: The following are equivalent

(i) The matrix inequalities $\mathbf{Q} > \mathbf{0}$ and $\mathbf{M} - \mathbf{R}\mathbf{Q}^{-1}\mathbf{R}^* > \mathbf{0}$ both hold.

(ii) The matrix inequality

$$\begin{bmatrix} \mathbf{M} & \mathbf{R} \\ \mathbf{R}^* & \mathbf{Q} \end{bmatrix} > \mathbf{0}.$$

is satisfied. For a proof, see Dullerud and Paganini (2002).

### Applications

- Finding a solution $\mathbf{x}$ to the LMI system $\mathbf{A}(\mathbf{x}) < 0$ is called *feasibility problem*
- Minimizing a convex objective function under some LMI constraints is also a convex problem. In particular, the linear objective minimization problem

$$\text{Minimize } \mathbf{c}^{\mathrm{T}}\mathbf{x}, \text{ subject to } \mathbf{A}(\mathbf{x}) < 0 \tag{7}$$

  plays an important role in LMI based design. This problem is called *eigenvalue problem*.
- The generalized eigenvalue problem

$$\text{Minimize } \lambda \text{ subject to } \left\{ \begin{array}{c} \mathbf{A}(\mathbf{x}) < \lambda \mathbf{B}(\mathbf{x}) \\ \mathbf{B}(\mathbf{x}) > 0 \\ \mathbf{C}(\mathbf{x}) < 0 \end{array} \right\} \tag{8}$$

  is another example.
- **Lyapunov's stability theorem**: A linear time-variant system given by $\dot{\mathbf{q}} = \mathbf{A}\mathbf{q}$ is stable if every eigenvalue of $\mathbf{A}$ has a negative real part. A matrix $\mathbf{A}$ satisfying this condition is called *Hurwitz*. A convenient way to check the eigenvalues of $\mathbf{A}$ is to employ the Lyapunov equation. This can be stated as follows:

  > All eigenvalues of $\mathbf{A}$ have negative real parts if and only if, for any given matrix $\mathbf{Q} = \mathbf{Q}^{\mathrm{T}} > 0$, there exists a unique solution $\mathbf{S} = \mathbf{S}^{\mathrm{T}} > 0$ for the Lyapunov equation given by $\mathbf{A}\mathbf{S} + \mathbf{S}\mathbf{A}^{\mathrm{T}} = -\mathbf{Q}$ (or $\mathbf{A}^{\mathrm{T}}\mathbf{S} + \mathbf{S}\mathbf{A} = -\mathbf{Q}$).

  This condition can also be stated as follows:

  > The system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$ is stable (and $\mathbf{A}$ is Hurwitz) if the LMI $\mathbf{A}\mathbf{S} + \mathbf{S}\mathbf{A}^{\mathrm{T}} + \mathbf{Q} \le 0$, or its strict version $\mathbf{A}\mathbf{S} + \mathbf{S}\mathbf{A}^{\mathrm{T}} + \mathbf{Q} < 0$, has a feasible solution $\mathbf{S} = \mathbf{S}^{\mathrm{T}} > 0$.
- LQG performance for a stable LTI system

$$\mathbf{G}\left\{ \begin{array}{c} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\boldsymbol{\omega} \\ \mathbf{y} = \mathbf{C}\mathbf{x} \end{array} \right\} \tag{9}$$

  where $\boldsymbol{\omega}$ is a white noise disturbance with unit covariance, the LQG or $H_2$ performance $\|\mathbf{G}\|_2$ is defined by

$$\|\mathbf{G}\|_2^2 := \int_{-\infty}^{\infty} Tr[\mathbf{G}^H(j\omega)\mathbf{G}(j\omega)]d\omega\,. \tag{10}$$

Here $\mathbf{G}(j\omega)$ represents the transfer function of the system. It can be shown that

$$\|\mathbf{G}\|_2^2 = \inf\{Tr[\mathbf{C}\mathbf{P}\mathbf{C}^{\mathrm{T}}]\} \quad \text{subject to} \quad \mathbf{A}\mathbf{P} + \mathbf{P}\mathbf{A}^{\mathrm{T}} + \mathbf{B}\mathbf{B}^{\mathrm{T}} < 0\,. \tag{11}$$

Here $\|\mathbf{G}\|_2^2$ is the global minimum of the LMI problem. Proof can be found in Dullerud and Paganini (2002).

## MATLAB LMI TOOLBOX

MATLAB LMI Toolbox provides a set of convenient functions to solve problems involving LMIs. In this section, we review some of these functions and give sample codes.

In general a problems involving LMIs are solved in two stages in MATLAB. First, we define the LMIs in the problem. This stage includes the specification of the decision variables in the LMIs and defining the LMIs based on these decision variables. As discussed in the previous section, there are several representations of an LMI. MATLAB conveniently utilizes the matrix form of decision variables given by (2) instead of the vector form of decision variables given by (1). In the second stage, the problem is solved numerically using the available solvers. If the problem includes minimization of a function with a vector form of decision variables, one need to convert the matrix form of decision variables into the vector form using additional functions.

### Specifying a System of LMIs

- We start with initializing LMI description. This is simply done by the statement

  ```
  setlmis([]);
  ```

  Note that this function does not take any parameter.
- Second we define the decision variables using the function `lmivar`.

  For example consider and LMI $\mathbf{C}\mathbf{X}\mathbf{C}^{\mathrm{T}} < 0$. Here, $\mathbf{C}$ is a constant matrix and $\mathbf{X}$ is the matrix of decision variables. The function `lmivar` allows defining several forms of decision matrix. As an example, let $\mathbf{X}$ be a symmetric matrix with block diagonal structure i.e.

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 & & & \\ & \mathbf{X}_2 & & \\ & & \ddots & \\ & & & \mathbf{X}_n \end{bmatrix}. \tag{12}$$

In this case, $\mathbf{X}$ has $n$ diagonal blocks. In this example let $n = 4$ and the dimensions of the matrices $\mathbf{X}_i$ be $5,3,1,2$, all are nonzero. $\mathbf{X}$ is defined by the following command:

```
structureX = [5,1;3,1;1,0;2,1]
X = lmivar(1,structureX);
```

Here the first input `1` states that `X` is a symmetric matrix with block diagonal. The row number of `structureX` states that `X` has 4 blocks. The first element of r-th row gives the dimension of the r-th block in `X`. The second element of r-th block defines the type of the block: `1` for full, `0` for scalar, `-1` for zero block.

- In the next step we define the LMIs one by one. For example consider the LMI given in the previous step: $\mathbf{CXC}^{\mathrm{T}} < 0$. In this case $\mathbf{C}$ is a $11 \times 11$ constant matrix. We define this LMI by

```
typeLMI1 = [1 1 1 1];
lmiterm(typeLMI1,C,C');
```

  As can be seen, in its simplest form, `lmiterm` takes three arguments. The first argument is a $1 \times 4$ vector. The first entry of this vector indicates which LMI is defined. In this case the first LMI is defined. Second and third entry is the position of the term being defined. This is detailed below with a better example. The fourth entry indicates which LMI decision variable is involved. In this example, since we have only one decision matrix, $\mathbf{X}$, this value is simply one. The second and third arguments of this function are the left and right multiplier of the decision matrix. If we wish to define the LMI $\mathbf{CXC}^{\mathrm{T}} > 0$, we simply use

```
typeLMI1 = [-1 1 1 1];
lmiterm(typeLMI1,C,C');
```

- Lets consider now a more involved example. Consider the following set LMIs:

$$\begin{bmatrix} \mathbf{CXC}^{\mathrm{T}} + \mathbf{B}^{\mathrm{T}}\mathbf{YA} & \mathbf{XF} \\ \mathbf{F}^{\mathrm{T}}\mathbf{X} & \mathbf{Y} \end{bmatrix} < 0 \quad \text{and} \quad \mathbf{DXD}^{\mathrm{T}} > 0 \tag{13}$$

In this case, we have two decision matrices, $\mathbf{X}$ and $\mathbf{Y}$ and two LMIs. Let $\mathbf{Y}$ be a full symmetric matrix of dimension 4. The MATLAB code will be

```
structureX = [5,1;3,1;1,0;2,1]
```

```
X = lmivar(1,structureX);
structureY = [4,1]
Y = lmivar(1,structureY);

lmiterm([1 1 1 1],C,C');   % term CXC'
lmiterm([1 1 1 2],B',A);   % term B'YA
lmiterm([1 1 2 1],1,F);   % term XF
lmiterm([1 2 1 1],F',1);   % term F'X
lmiterm([1 2 2 2],1,1);   % term Y

typeLMI2 = [-1 1 1 1];
lmiterm(typeLMI2,D,D');   % term DXD'
```

In this example, it is easily observed that the second and third term of the first parameter of `lmiterm` define the locations of the corresponding term in the LMI.

- Functions defined above are capable of defining more complex LMIs. See LMI Toolbox manual for the details.
- The final step in defining the LMIs is to create an LMI object using the following command:

```
myLMIsystem = getlmis;
```

### Solving an LMI problem

- Matlab can solve three types of LMI problems: feasibility, eigenvalue and generalized eigenvalue problems. Here we show an eigenvalue problem as an example. Consider the eigen value problem:

$$\min_{\mathbf{X},\mathbf{Y}} Tr(\mathbf{X}) + Tr(\mathbf{B}^{\mathrm{T}}\mathbf{Y}\mathbf{A}) \quad \text{subject to inequalities (13).} \tag{14}$$

This problem is not exactly in the form of (7) but can be represented in that form using some algebra i.e., the elements of $\mathbf{X}$ and $\mathbf{Y}$ can be represented as a vector $\mathbf{x}$, and similarly one can find a $\mathbf{c}$ such that $Tr(\mathbf{X}) + Tr(\mathbf{B}^{\mathrm{T}}\mathbf{Y}\mathbf{A}) = \mathbf{c}^{\mathrm{T}}\mathbf{x}$ In MATLAB, we use the term $Tr(\mathbf{X}) + Tr(\mathbf{B}^{\mathrm{T}}\mathbf{Y}\mathbf{A})$ and employ some additional functions to find $\mathbf{c}$ and define $\mathbf{x}$.

- First, we find the size of $\mathbf{x}$ using

```
nx = decnbr(myLMIsystem);
```

- Second, we initialize the vector $\mathbf{c}$

```
c = zeros(nx,1);
```

Note that dimensions of $\mathbf{c}$ and $\mathbf{x}$ are same.

- Then, the terms $Tr(\mathbf{X}) + Tr(\mathbf{B}^{\mathrm{T}}\mathbf{Y}\mathbf{A})$, $\mathbf{c}$ and $\mathbf{x}$ are defined as follows:

```
for i = 1:n
  [Xi, Yi] = defcx(myLMIsystem,i,X,Y);
  c(i) = trace(Xi) + trace(B'*Yi*A);
end
```

Here, the function `defcx` simply associates the elements of $\mathbf{X}$ and $\mathbf{Y}$ to $\mathbf{x}$ and finds the corresponding vector $\mathbf{c}$.

- Next, we solve the problem with the MATLAB solver

```
[minval, xopt] = mincx(myLMIsystem,c,[1e-5,0,0,0,0]);
```

Here the last input is a vector for options of the solver. The output `minval` is the minimum value of $Tr(\mathbf{X}) + Tr(\mathbf{B}^{\mathrm{T}}\mathbf{Y}\mathbf{A})$ and `xopt` is the corresponding $\mathbf{x}$.

- In the following step, we obtain the matrices $\mathbf{X}$ and $\mathbf{Y}$ using

```
Xopt = dec2mat(myLMIsys,xopt,X);
Yopt = dec2mat(myLMIsys,xopt,Y);
```

This concludes the LMI solver usage.

### An Example

In this section, an eigevalue problem and its MATLAB implementation is given. This problems is he LMI version of a standard linear quadratic regulator (LQR) problem.

- The problem LMI-LQR problem:

$$\begin{aligned} &\min_{\mathbf{Y},\mathbf{S},\mathbf{X}} \; Tr(\mathbf{QS}) + Tr(\mathbf{X}) + Tr(\mathbf{YN}) + Tr(\mathbf{N}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}) \\ &\text{subject to } \mathbf{AS} - \mathbf{BY} + \mathbf{SA}^{\mathrm{T}} - \mathbf{Y}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}} + \mathbf{EE} < 0, \quad \text{and} \quad \begin{bmatrix} \mathbf{X} & \mathbf{R}^{1/2}\mathbf{Y} \\ \mathbf{Y}^{\mathrm{T}}\mathbf{R}^{1/2} & \mathbf{S} \end{bmatrix} > 0 \end{aligned} \tag{15}$$

- MATLAB Code

```
function K = lmilqr (sys, lq, options)

%LMILQR solves the standard LQR problem (type "help lqr" for more info)
%  using LMI techniques.
%
%   The problem statement is as follows:
```

```
%
%   Consider a LTI system with a state-space representation given by
%
%       q = A q + B u + E w
%       z = Czq + Dzu + Ezw
%
%   Standard LQR problem calculates a optimal gain matrix K, such
%   that the state-feedback law u = -Kq minimizes the cost function
%
%        J = Integral {x'Qx + u'Ru + 2*x'Nu} dt
%                                .
%    subject to the state dynamics  x = Ax + Bu + Ew.
%
%   This problem is equivalent to the following eigenvalue problem:
%
%      min   Tr(PQ) + Tr(X) - Tr(YN) - Tr(N'Y')
%    (Y,P,X)
%
%   subject to  AP - BY + PA' - Y'B' + EE' < 0
%
%              [ X        Sqrt(R)Y ]
%              [                   ]  > 0
%              [Y'Sqrt(R)     P    ]
%
%   LMILQR solves the above eigenvalue problem to compute the optimal
%   gain matrix given by K = Y inv(P).
%   It uses the "mincx" function of the LMI Control Toolbox.
%
%   SYNTAX
%
%   K = lqrlmi (sys, lq, options)
%       where sys is a structure with sys.A, sys.B and sys.E matrices and
%       lq is a structure with lq.Q, lq.R and lq.N matrices.
%
%       options is defined by the options of "mincx" function. If ignored
%       it is assumed as [1e-20, 100, -1, 5, 1]
%
%   written by Baris Erkus
%   ver 1.0

% Resolve the inputs
A = sys.A; B = sys.B; E = sys.E;
Q = lq.Q; R = lq.R; N = lq.N;

nstate = size(A,1);

% Enforce symmetry of Q and R in case they are different due to numerical
% roundoff in computation
Q = (Q+Q.')/2;
R = (R+R.')/2;
R_half = R^(1/2);
R_half=(R_half+R_half.')/2;

%%
```

```
%% Defintion of LMIs
%%

setlmis([]);
P=lmivar(1, [nstate,1]);
X=lmivar(2, [1,1]);
Y=lmivar(2, [1,nstate]);

% Subject function, LMI #1
%            AP - BY + PA' - Y'B' + EE' < 0

lmiterm([1 1 1 P], A,  1, 's');               % LMI #1:  AP + PA'
lmiterm([1 1 1 Y], B, -1, 's');               % LMI #1: -BY - Y'B'
lmiterm([1 1 1 0], E*E');                     % LMI #1:  EE'

% Subject function, LMI #2:
%            [ X       Sqrt(R)Y ]
%            [                  ]  > 0
%            [Y'Sqrt(R)     P    ]

lmiterm([-2 1 1  X], 1, 1);                   % LMI #2: X
lmiterm([-2 2 1 -Y], 1, R_half);              % LMI #2: Y'*sqrt(R)
lmiterm([-2 2 2  P], 1, 1);                   % LMI #2: P

% Create the LMI system
lmisys = getlmis;

% Defining vector "c"
n = decnbr(lmisys);
c = zeros(n,1);
for i=1:n
   [Pj, Xj, Yj] = defcx(lmisys, i, P, X, Y);
   c(i) = trace(Q*Pj) + trace(Xj) - trace(Yj*N) - trace(N'*Yj');
end

% Solving LMIs
if (nargin == 2);
    options = [1e-20, 100, -1, 5, 1];
end;
[copt, xopt] = mincx (lmisys, c, options);

% Results
Xopt = dec2mat(lmisys, xopt, X);
Yopt = dec2mat(lmisys, xopt, Y);
Popt = dec2mat(lmisys, xopt, P);

K = Yopt*inv(Popt);
```

## REFERENCES

Boyd, S., Ghaoui, L. E., Feron, E., and Balakrishnan, V. (1994). *Linear Matrix Inequalities in System and Control Theory*, Studies in Applied Mathematics, **15**, SIAM, Philadelphia, Pennsylvania.

Dullerud, G. E., and Paganini, F. (2000). *A Course in Robust Control Theory: A Convex Approach*, Springer, New York.

Gahinet, P., Nemirovski, A., Laub., A. J., and Chilali, M. (1995). *LMI control toolbox*. MathWorks, Natick, Massachusetts.

Johnson E. A., and Erkus, B. (2003). "Dissipative control of structures with smart dampers via LMI synthesis" *in preparation*.