EE363 Review Session 4:
Linear Matrix Inequalities

In this review session we'll go over some examples on linear matrix inequalities (LMIs). We'll also introduce a software package called `cvx`, which you can use to solve semidefinite programs (SDPs), and LMIs. (You'll need to use this for some homework and exam problems.) If you haven't done so already, please read over the SDP tutorial in the support notes section on the course website.

Linear Matrix Inequalities

Recall from lectures that a linear matrix inequality (LMI) in the variable $x \in \mathbf{R}^n$ has the form

$$F(x) = F_0 + x_1 F_1 + \cdots + x_n F_n \geq 0,$$

where $F_0 \in \mathbf{R}^{m\times m}, \ldots, F_n \in \mathbf{R}^{m\times m}$ are symmetric matrices.

Many inequalities can be represented as LMIs. For example, you've seen in lectures that a set of linear inequalities

$$a_i^T x \leq b_i, \quad i = 1, \ldots, k,$$

can be represented as the LMI

$$\begin{bmatrix} b_1 - a_1^T x & 0 & \cdots & 0 \\ 0 & b_2 - a_2^T x & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & b_k - a_k^T x \end{bmatrix} \geq 0.$$

Example: How can we represent the linear equality constraints

$$a_i^T x = b_i, \quad i = 1, \ldots, k,$$

with variable x, as an LMI?

Solution. We can write the constraint as

$$a_i^T x \leq b_i, \quad a_i^T x \geq b_i, \quad i = 1, \ldots, k,$$

which is equivalent to the LMI

$$\mathbf{diag}\left(b_1 - a_1^T x, \ldots, b_k - a_k^T x, a_1^T x - b_1, \ldots, a_k^T x - b_k\right) \geq 0.$$

Another example is an LMI that arises frequently in Lyapunov theory,

$$A^T P + PA + Q \leq 0,$$

where P is the variable (why is this an LMI?). In this case, we can solve the LMI analytically, by solving the Lyapunov equation $A^T P + PA + Q = 0$.

Now let's take a look at a useful result which you are asked to derive in this week's homework. Consider a matrix

$$X = \left[\begin{array}{cc} A & B \\ B^T & C \end{array} \right] \in \mathbf{R}^{n \times n},$$

where $A \in \mathbf{R}^{k \times k}$. We want to show that if $A > 0$, then $X \geq 0$ if and only if $S = C - B^T A^{-1} B \geq 0$. (This result allows us to represent Schur complements as LMIs; you'll see very soon that this has lots and lots of applications.)

To show this, we recall from homework 1 that

$$\inf_u \left[\begin{array}{c} u \\ v \end{array} \right]^T \left[\begin{array}{cc} A & B \\ B^T & C \end{array} \right] \left[\begin{array}{c} u \\ v \end{array} \right] = v^T (C - B^T A^{-1} B) v.$$

Thus, if $C - B^T A^{-1} B \geq 0$, then $v^T (C - B^T A^{-1} B) v \geq 0$ for all $v \in \mathbf{R}^{n-k}$. This implies that the above quadratic form must be nonnegative for all $u \in \mathbf{R}^k$ and $v \in \mathbf{R}^{n-k}$, so $X \geq 0$. Similarly, if $X \geq 0$, then the quadratic form is nonnegative for all u and v, which implies that the infimum over u must also be nonnegative, for any v. Thus, $v^T (C - B^T A^{-1} B) v \geq 0$ for all $v \in \mathbf{R}^{n-k}$, and so $C - B^T A^{-1} B \geq 0$.

Let's take a look at a few applications of this result.

Example: Represent the inequality

$$\|Ax - b\| \leq \gamma,$$

with $\gamma > 0$ and variable x, as a linear matrix inequality.

Solution. The inequality is equivalent to $\|Ax - b\|^2 \leq \gamma^2$, which can be written as

$$\gamma^2 - (Ax - b)^T (Ax - b) \geq 0.$$

Using the Schur complement result we just derived, we can write this as

$$\left[\begin{array}{cc} I & Ax - b \\ (Ax - b)^T & \gamma^2 \end{array} \right] \geq 0.$$

Example: Represent the inequalities

$$P \leq A^T PA + Q - A^T PB(R + B^T PB)^{-1} B^T PA, \quad P \geq 0$$

where $R = R^T > 0$, as a single linear matrix inequality (in variable P). This expression looks very similar to the steady state discrete-time Riccati equation we encountered in lecture 3, except that the equality is relaxed to an inequality. (In fact, using this LMI representation of the Riccati inequality, we can solve the Riccati equation by solving a semidefinite program.)

Solution. These inequalities can be written as

$$0 \leq A^T P A + Q - P - A^T P B (R + B^T P B)^{-1} B^T P A, \quad P \geq 0$$

which is equivalent to the LMI

$$\begin{bmatrix} R + B^T P B & B^T P A & 0 \\ A^T P B & A^T P A + Q - P & 0 \\ 0 & 0 & P \end{bmatrix} \geq 0.$$

Example: Matrix norm inequalities. Represent the inequality

$$\|A\| \leq \gamma,$$

with $\gamma > 0$ and variable A, as a linear matrix inequality.

Solution. The inequality $\|A\| \leq \gamma$ is equivalent to $\lambda_{\max}(A^T A) \leq \gamma^2$, which can be written as

$$A^T A \leq \gamma^2 I.$$

This can be expressed as the LMI

$$\begin{bmatrix} I & A \\ A^T & \gamma^2 I \end{bmatrix} \geq 0.$$

Semidefinite programs

A semidefinite program (SDP) is an optimization problem where the objective is a linear functions of the variables, and the constraints consist of LMI constraints, and linear equality constraints:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & F_0 + x_1 F_1 + \cdots + x_n F_n \geq 0 \\ & Ax = b. \end{array}$$

(Here, x is the variable.) We just showed that a set of linear equality constraints can be represented as an LMI, so an SDP is really just an optimization problem with a linear objective and LMI constraints. The form we've given you is just a standard representation of SDPs. You'll learn more about SDPs next quarter if you take convex optimization. For now, you need to make sure you understand how to transform various problems into an SDP format, and solve them using `cvx`. For example, to find a matrix $A \in \mathbf{R}^{n \times n}$ that satisfies the inequality

$$\|A\| \leq \gamma,$$

and maximizes $\mathbf{Tr}(A)$, we can solve the SDP

$$\begin{array}{ll} \text{maximize} & \mathbf{Tr}(A) \\ \text{subject to} & \left[\begin{array}{cc} I & A \\ A^T & \gamma^2 \end{array} \right] \geq 0. \end{array}$$

We can use `cvx` to solve this problem by entering a `cvx` SDP specification into a Matlab script (or even just the Matlab command prompt). The `cvx` specification for this problem is

```
cvx_begin sdp
    variable A(n,n)
    [eye(n), A; A', gamma^2*eye(n)] >= 0
    maximize (trace(A))
cvx_end
```

(We assume that `n` and `gamma` are already defined.) In this class, we'll work exclusively in SDP mode, which means that we begin every `cvx` specification with `cvx_begin sdp`. Those of you who have taken EE364A and are comfortable with convex optimization can use `cvx` however you like. If `cvx` is new to you, we recommend that you stick with SDP mode, and represent all your optimization problems in SDP form. After `cvx_begin sdp`, we define the variable, `A`, which is an n-by-n matrix. Obviously, we must define the variables first, before defining our objective and constraints. Then on the third line, we specify our LMI constraint, and on the fourth line we define our objective. Finally, we end a `cvx` specification with `cvx_end`.

When this code segment is processed by Matlab, an SDP is formed, and a package called SDPT3 is used to solve the SDP. If the SDP is solved (*i.e.*, an A is found that satisfies the constraints), then `A` becomes an ordinary numerical matrix. Otherwise, if the problem is infeasible, the entries of `A` are set to `NaN`. `cvx` will also output to screen lots and lots of information (unless you explicitly disable it). Most of this information shows the progress of the optimization algorithm (which you can ignore), but right at the end of the deluge, it should tell you the final status: *e.g.*, `Solved`, `Infeasible`, *etc.*, and also the optimal objective value (which is set to `Inf` if the problem is infeasible). You can also access the status and optimal value by examining the variables `cvx_status` and `cvx_optval`.

As you can see, `cvx` is really very straightforward to use. There are however, a few caveats we should mention. First of all, `cvx`, and most other optimization solvers, do *not* handle strict inequalities. For example, the operator `>` is the same as `>=`, and implements the nonstrict inequality: $\geq$. If you need to implement a strict inequality, make sure you use the methods we mention in lecture 14 (slide 9).

Another common mistake while using `cvx` is in confusing matrix inequalities with elementwise inequalities. In SDP mode every inequality is interpreted as a matrix inequality. For instance, if you want to impose the inequality $A_{ij} \leq 1$, the statement

```
A <= ones(n,n)
```

is *incorrect*, since the inequality will be interpreted as a matrix inequality in SDP mode. Instead you should use the statement

```
diag(vec(A)) <= eye(n^2)
```

Here the function `vec` vectorizes the matrix `A`. Actually, this representation of the constraint is a little pedantic. In SDP mode inequalities involving vectors are still interpreted elementwise, so in fact the statement `vec(A) <= ones(n^2,1)` works as well. If you want to be safe however, you should just convert all your constraints into LMIs.

Let's do an example involving strict LMIs. Suppose you want to find a symmetric matrix P that satisfies

$$A_1^T P + P A_1 < 0, \quad A_2^T P + P A_2 < 0, \quad P > 0.$$

One interpretation here is that if a P exists, then the Lyapunov function $V(z) = z^T P z$ proves that the system $\dot{x} = A(t)x$, where $A(t) \in \{A_1, A_2\}$, is globally asymptotically stable. You saw something similar to this in homework 6, problem 6 (except you couldn't guarantee that such a P existed). To solve the problem, notice that the LMIs are homogeneous in P, which means that they are equivalent to the non-strict LMIs

$$A_1^T P + P A_1 \leq -I, \quad A_2^T P + P A_2 \leq -I, \quad P \geq I.$$

(Make sure you understand why this is the case.) The following `cvx` code solves this problem.

```
cvx_begin sdp
    variable P(n,n) symmetric
    A1'*P+P*A1 <= -eye(n)
    A2'*P+P*A2 <= -eye(n)
    P >= eye(n)
cvx_end
```

Here, there is no need to add an objective to the `cvx` specification. We could add `minimize(0)`, or in fact any other linear objective, but we don't need to.

Example: Now let's do an example that uses many of the concepts we've seen so far. Suppose we are given a matrix $M \in \mathbf{R}^{n \times n}$. How can we find an nonsingular diagonal matrix $D \in \mathbf{R}^{n \times n}$ such that

$$\|DMD^{-1}\| \leq \gamma,$$

where $\gamma > 0$?

Solution. The inequality is equivalent to

$$(DMD^{-1})^T (DMD^{-1}) \leq \gamma^2 I,$$

which can be written as

$$D^{-1} M^T D^2 M D^{-1} \leq \gamma^2 I.$$

Since D is nonsingular, this is the same as

$$M^T E M \leq \gamma^2 E,$$

where $E = D^2 > 0$. Thus, to find D, we solve the LMIs

$$M^T E M \leq \gamma^2 E, \quad E > 0$$

to find E. Then we simply take $D = E^{1/2}$. To solve this using `cvx`, we notice that both LMIs are homogeneous in E, so we can represent them as non-strict LMIs

$$M^T E M \leq \gamma^2 E, \quad E \geq I.$$

The following `cvx` code solves the problem.

```
cvx_begin sdp
    variable E(n,n) diagonal
    M'*E*M <= gamma^2*E
    E >= eye(n)
cvx_end
```

Finally, let's do a Lyapunov theory example from the lectures. Suppose we have the following nonlinear system,

$$\dot{x} = Ax + g(x), \quad \|g(x)\| \leq \gamma \|x\|,$$

and we'd like to find a $P > 0$ such that $V(z) = z^T P z$ satisfies

$$\dot{V}(x) \leq -\alpha V(x), \text{ for all } x$$

where $\alpha > 0$ is given. If such a P exists, then $V(z) = z^T P z$ proves that the system is globally asymptotically stable. From lecture 14 (slide 28), we know that necessary and sufficient conditions for the existence of a quadratic Lyapunov function are the LMIs

$$P \geq I, \quad \begin{bmatrix} A^T P + P A + \alpha P + \tau \gamma^2 I & P \\ P & -\tau I \end{bmatrix} \leq 0,$$

in variables P and τ. We can solve these LMIs using the following `cvx` code.

```
cvx_begin sdp
    variable P(n,n) symmetric
    variable tau
    P >= eye(n)
    [A'*P+P*A+alpha*P+tau*gamma^2*eye(n), P; P, -tau*eye(n)] <= 0
cvx_end
```