Advanced Control for Robotics - Homework 6

涂志鑫 12131094

2022.05

Problem 1

$\textbf{Schur complement lemma}$: Find an equivalent semidefinite condition of the form $G(x) \succeq 0$ for each of the following statements. Make sure the matrix G(x) you obtained is affine w.r.t. x, where x is a vector or matrix variable of appropriate dimension. Please show your steps.

(a) (Singular value bound): $\sigma(A(x)) < \beta$, where $A : R^n \to R^{q\times m}$ is affine in x ∈ Rn and σ(·) denotes the singular value of a matrix.

(b) (Riccati inequality): $A^Tx + xA + xBR^{-1}B^Tx + Q \prec 0$, with $x \in S^n_{++}$, $R \in S^p_{++}$, $Q \in S^n$ and $A \in R^{n\times n}$, and $B \in R^{n\times p}$.

Solution:

(a) $\textbf{(Singular value bound)}$ We can know the sigular value of a matrix M is $\sigma(M) = \sqrt{eig(M^TM)}$.

For the bound of the singular value, $\sigma(A(x)) < \beta$ can be expressed by

$$\begin{aligned} \sqrt{eig(A(x)^TA(x))} &< \beta \\ eig(A(x)^TA(x)) &< \beta^2 \\ \beta^2 - max\, eig(A(x)^TA(x)) &> 0 \end{aligned}$$

Therefore, $\beta^2I - A(x)^TA(x)$ is PD, For the Schur complement lemma, the inequality can be

$$\begin{aligned} \beta^2I - A(x)^TA(x) &> 0 \\ G(x) = \begin{bmatrix} I & A(x) \\ A^T(x) & \beta^2I \end{bmatrix} &> 0 \end{aligned}$$

$$G(x) = \begin{bmatrix} I & 0 \\ 0 & \beta^2I \end{bmatrix} + \begin{bmatrix} 0 & A(x) \\ A^T(x) & 0 \end{bmatrix} > 0$$

Since $A(x)$ is affine w.r.t x, so $G(x)$ is also affine w.r.t x.

(b) $\textbf{(Riccati inequality)}$ $A^Tx + xA + xBR^{-1}B^Tx + Q \prec 0$ For $x \in S^n_{++}$, $R \in S^p_{++}$, $Q \in S^n$, $A \in R^{n\times n}$ and $B \in R^{n\times p}$.

The the inequality can be expressed

$$A^Tx + xA + xBR^{-1}B^Tx + Q = 2xA + Q + xBR^{-1}B^Tx \prec 0$$

$$-(2xA + Q) - xBR^{-1}B^Tx > 0$$

For the Schur complement lemma, the inequality can change to

$$M = \begin{bmatrix} R & B^T x \\ xB & -Q - 2xA \end{bmatrix} > 0$$

$x \in S_{++}^n$ can be expressed by

$$x = \sum_i \sum_{j \le i} x_{ij} S_{ij}, S_{ij} \in S^n$$

$$\begin{aligned} M &= \begin{bmatrix} R & B^T x \\ xB & -Q - 2xA \end{bmatrix} \\ &= \begin{bmatrix} R & 0 \\ 0 & -Q \end{bmatrix} + \sum_i \sum_{j \le i} x_{ij} \begin{bmatrix} 0 & B^T S_{ij} \\ S_{ij} B & -2S_{ij} A \end{bmatrix} > 0 \end{aligned}$$

For $B \in R^{n \times p}, S_{ij} \in S^n$, then $B^T S_{ij} = S_{ij} B$, $\begin{bmatrix} 0 & B^T S_{ij} \\ S_{ij} B & -2S_{ij} A \end{bmatrix} \in S^{n+p}$.

Therefore, $F_0 = \begin{bmatrix} R & 0 \\ 0 & -Q \end{bmatrix}$,$F(x) = \sum_i \sum_{j \le i} x_{ij} \begin{bmatrix} 0 & B^T S_{ij} \\ S_{ij} B & -2S_{ij} A \end{bmatrix}$.

$G(x) = F_0 + F(x) > 0$ is affine w.r.t. x.

Problem 2

Ellipsoid : Ellipsoid in R^n have two equivalent representations: (i) $E_1(P, x_c) = x \in R^n : (x - x_c)P^{-1}(x - x_c) \le 1$ and (ii) $E_2(A, x_c) = Au + x_c :\| u \|^2 \le 1$. The second representation can be derived from the first by letting $A = P^{\frac{1}{2}}$. Given $E_1(P, xc)$ with $P \in S_{++}^n$, its volume is $\nu_n \sqrt{det(P)}$ where ν_n is the volume of unit ball in R^n, its semi-axes directions are given by the eigenvectors of P and the lengths of semi-axes are $\sqrt{\lambda_i}$, where $\sqrt{\lambda_i}$ are eigenvalues of P.

(a) Given a half space $x \in Rn : a^T x \le 1$. Show that the Ellipsoid $E_2(A, 0)$ is contained in the eigenvectors of P and the lengths of semi-axes are the half space if and only if $a^T AA^T a \le 1$.

(b) Note that for any $P \in S_{++}^n$, the function $log(det(P))$ is concave in the matrix variable P. Formulate a convex optimization problem to find the matrix $P \in S_{++}^n$ such that $E_1(P, 0)$ is the largest ellipsoid contained in the polyhedron $x \in R^n : a_i^T x \le 1, i = 1, \dots, m$

(c) Use Drake to solve the above problem with $a_1^T = [-1, 1]$, $a_2^T = [2, -1]$, $a_3^T = [1, 3]$, $a_4^T = [-2, -5]$. Visualize the polyhedron region and your ellipsoid solution (you can use Matlab for the visualization if you prefer matlab)

Solution:

(a) The ellipsoid $E_2(A, 0) = \{Au : ||u||^2 \le 1\}$, so the ellipsoid $x = Au, ||u||^2 \le 1$。

For the half-space $x \in Rn : a^T x \le 1$, in order to get the ellipsoid contained in the half space, for all $x \in R^n$ in the ellipsoid, the expression $x = Au$ must satisfy the inequality $a^T x \le 1$.

Plug-in the $x = Au$ to the ineuality we get

$$
\begin{aligned}
a^T Au &\le 1 \\
||a^T Au||^2 &\le 1 \\
(a^T Au)(a^T Au)^T &\le 1 \\
a^T Auu^T Aa &\le 1
\end{aligned}
$$

Since we know the $||u||^2 \le 1$, $uu^T - I \le 0$, so

$$a^T AAa \le 1$$

(b) The volume of a ellipsoid $E_1(P, x_c) = x \in R^n : (x - x_c)P^{-1}(x - x_c) \le 1$ is $V = \nu_n(\sqrt{det(P)})$. To find the largest epllipsoid, that is to find a P which maximize $log(\sqrt{det(P)})$.

We use the second representation to represent the ellipsoid $E_2(A, x_c) = Au + x_c :\| u \|^2 \le 1$, for $A = P^{\frac{1}{2}}$, it can be $E = P^{\frac{1}{2}} u + x_c :\| u \|^2 \le 1$. ((The volume of ellipsoid is proportional to $det(P) = det(A^2)$, so the problem can be formulated by

$$
\begin{cases}
\max_{P,\tau,\alpha} & log(det(A)) \\
\text{subject.:} & sub_{||u||^2 \le 1} a_i^T (Au + x_c) \le 1, i = 1, 2, \dots, m,
\end{cases}
$$

The constraint can be equvilent to

$$
\begin{cases}
\max_{P,\tau,\alpha} & log(det(A)) \\
\text{subject.:} & ||Aa_i||_2 + a_i^T x_c \le 1, i = 1, 2, \dots, m,
\end{cases}
$$

Since $log(det(A))$ is concave, and the constraint condition is affine function, the problem is a convex optimization problem.

(c) Using matlab CVX toolbox to solve the given problem, the code is shown below.

```
n = 2;
px = [-0.5 4/7 1/3 -6/7];
py = [0.5 1/7 -1/3 1/7];
m = 4;
px = [px px(1)];
py = [py py(1)];
A = zeros(m,n); b = zeros(m,1);
A = [-1,1;2,-1;1,3;-2,-5];
b = [1;1;1;1;1];

% formulate and solve the problem
cvx_begin
    variable B(n,n) symmetric
    variable d(n)
    maximize( det_rootn( B ) )
    subject to
       for i = 1:m
           norm( B*A(i,:)', 2 ) + A(i,:)*d <= b(i);
       end
cvx_end
```

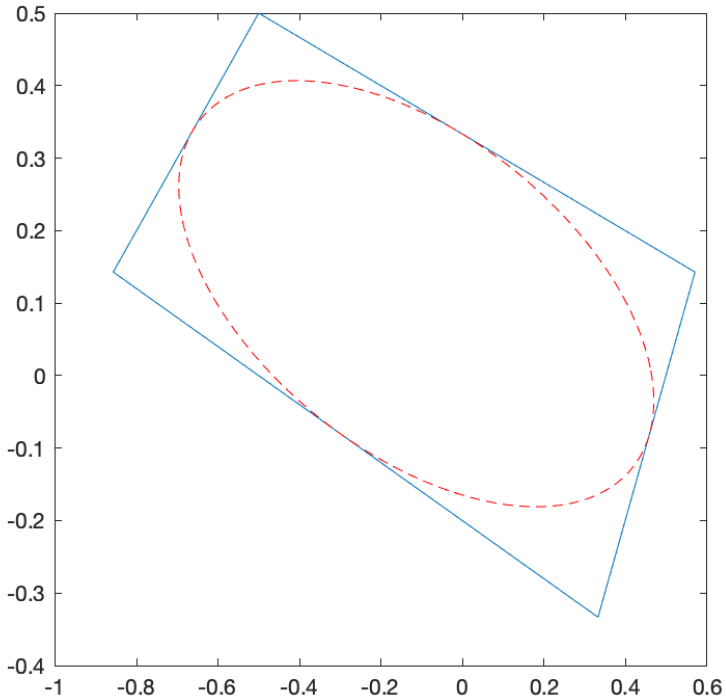
```
% make the plots
noangles = 400;
angles   = linspace( 0, 2 * pi, noangles );
ellipse_inner  = B * [ cos(angles) ; sin(angles) ] + d * ones(
1, noangles );

clf
plot(px,py)
hold on
plot( ellipse_inner(1,:), ellipse_inner(2,:), 'r--' );
axis square
hold off
```

The output of the code is

```
Calling SDPT3 4.0: 30 variables, 14 equality constraints
   For improved efficiency, SDPT3 is solving the dual problem.
------------------------------------------------------------

 num. of constraints = 14
 dim. of sdp    var  =  6,   num. of sdp  blk  =  2
 dim. of socp   var  = 12,   num. of socp blk  =  4
 dim. of linear var  =  5
*******************************************************************

   SDPT3: Infeasible path-following algorithms
*******************************************************************

 version  predcorr  gam  expon  scale_data
   HKM      1      0.000   1        0
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj
cputime
----------------------------------------------------------------
---
 0|0.000|0.000|1.8e+01|1.2e+01|1.2e+03| 1.385641e+01
0.000000e+00| 0:0:00| chol  1  1
 1|1.000|0.741|7.8e-06|3.2e+00|3.3e+02| 3.397599e+01
-6.874381e+00| 0:0:00| chol  1  1
 2|1.000|0.988|5.5e-06|4.9e-02|2.7e+01| 2.311574e+01 -9.692387e-
02| 0:0:00| chol  1  1
 3|0.898|1.000|1.2e-06|1.0e-03|3.3e+00| 3.235468e+00 -2.201325e-
02| 0:0:00| chol  1  1
 4|0.821|1.000|2.4e-07|1.0e-04|6.7e-01| 7.185773e-01  4.730671e-
02| 0:0:00| chol  1  1
 5|0.775|1.000|5.7e-08|1.0e-05|2.7e-01| 5.394086e-01  2.693960e-
01| 0:0:00| chol  1  1
 6|0.947|0.999|3.7e-09|1.0e-06|1.5e-02| 3.906265e-01  3.758807e-
01| 0:0:00| chol  1  1
 7|0.975|0.979|5.9e-10|1.2e-07|3.4e-04| 3.849103e-01  3.845749e-
01| 0:0:00| chol  1  1
 8|0.946|0.936|6.6e-10|1.7e-08|2.1e-05| 3.847799e-01  3.847587e-
01| 0:0:00| chol  1  1
 9|1.000|1.000|1.2e-09|1.3e-10|2.8e-06| 3.847739e-01  3.847712e-
01| 0:0:00| chol  1  1
10|1.000|1.000|7.8e-13|2.0e-10|8.7e-08| 3.847731e-01  3.847730e-
01| 0:0:00| chol  1  1
```

```
11|1.000|1.000|6.8e-12|1.0e-12|3.8e-09| 3.847731e-01  3.847731e-
01| 0:0:00|
  stop: max(relative gap, infeasibilities) < 1.49e-08
-----------------------------------------------------------------
---
 number of iterations   = 11
 primal objective value =  3.84773066e-01
 dual   objective value =  3.84773062e-01
 gap := trace(XZ)       = 3.83e-09
 relative gap           = 2.16e-09
 actual relative gap    = 2.16e-09
 rel. primal infeas (scaled problem)   = 6.76e-12
 rel. dual     "        "       "      = 1.00e-12
 rel. primal infeas (unscaled problem) = 0.00e+00
 rel. dual     "        "       "      = 0.00e+00
 norm(X), norm(y), norm(Z) = 2.2e+00, 1.0e+00, 3.4e+00
 norm(A), norm(b), norm(C) = 1.4e+01, 2.0e+00, 3.0e+00
 Total CPU time (secs)  = 0.14
 CPU time per iteration = 0.01
 termination code       =  0
 DIMACS: 6.8e-12  0.0e+00  1.5e-12  0.0e+00  2.2e-09  2.2e-09
-----------------------------------------------------------------
---

------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +0.384773
```

The plot of the ellipsoid is



Problem 3

Stability of Lur'e system: Consider a nonlinear system $\dot{x} = Ax + b\phi(t, c^T x)$ where $A \in R^n \times n$, $b \in R^n$, $c \in R^n$, and for each time t, the function $\phi(t, \cdot) : R \to R$ satisfies sector nonlinearity $|\phi(t, y)| \leq \alpha |y|$ for all y (but is otherwise unknown). Such a system represents a very general class of control systems involving time-varying nonlinearities and/or uncertainties, and is often called a Lur'e problem. We would like to find a positive definite Lyapunov function $V(x) = x^T Px$ that satisfies $\dot{V}(x) \leq -\beta V(x)$ for all x, and for any function ϕ satisfying the inequality given above. You can assume that A, b, c, α, β are given.

(a) Explain how to find such a P (or determine that no such P exists) by expressing the problem as an LMI.

(b) Use CVX to construct such a Lyapunov function for the following instance of Lur'e problem:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -3 & -3 \end{bmatrix}, b = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} c = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \alpha = 0.7, \beta = 0.1$$

Solution:

(a) For the Lyapunov function $V(x) = x^T Px$, $\dot{V}(x) = \dot{x}^T Px + x^T P\dot{x} \leq -\beta V(x)$. The nonlinear system is $\dot{x} = Ax + b\phi(t, c^T x)$, assume $u = b\phi(t, c^T x)$, plug in the Lyapunov function, get

$$\dot{V}(x) = [Ax + b\phi(t, c^T x)]Px + x^T P[Ax + b\phi(t, c^T x)] \leq -\beta(x^T Px)$$

Because the fuction $\phi(t, \cdot)$ satisfies $|\phi(t, y)| \leq \alpha|y|$, the inequality above can be

$$\begin{aligned} {}^T Px + x^T P[Ax + u] + \beta(x^T Px) \leq 0 \\ x^T(a^T P + PA + \beta P)x + x^T Pu + u^T Px \leq 0 \end{aligned}$$

Change it to the matrix form, for all the $(x, \phi(t, c^T x))$,

$$\begin{bmatrix} x^T & \phi(t, c^T x)^T \end{bmatrix} \begin{bmatrix} -(A^T P + PA + \beta P) & -Pb \\ -b^T P & 0 \end{bmatrix} \begin{bmatrix} x \\ \phi(t, c^T x) \end{bmatrix} \succeq 0$$

For the function $\phi(t, \cdot) : R \to R$ satisfies sector nonlinearity $|\phi(t, y)| \leq \alpha|y|$, so $||\phi(t, c^T x)||^2 \leq \alpha^2(x^T cc^T x)$

Change it to matrix form, we can get

$$\begin{bmatrix} x^T & \phi(t, c^T x)^T \end{bmatrix} \begin{bmatrix} \alpha^2 cc^T & 0 \\ 0 & -I \end{bmatrix} \begin{bmatrix} x \\ \phi(t, c^T x) \end{bmatrix} \succeq 0$$

Assume $G_0 = \begin{bmatrix} -(A^T P + PA + \beta P) & -Pb \\ -b^T P & 0 \end{bmatrix}$, $G_1 = \begin{bmatrix} \alpha^2 cc^T & 0 \\ 0 & -I \end{bmatrix}$, by $\textbf{s-procedure}$, we have $\textcolor{red}{\text{\textbackslash exist}}\tau > 0$, s.t. $G_0 > \tau G_1$, then the problem can be formulated by

$$\begin{cases} \min_{P,\tau,\alpha} & -\beta \\ \text{subject.:} & P \succ 0, \\ & \tau > 0, \\ & G_0 - \tau G_1 \succeq 0 \end{cases}$$

In [ ]:

```
# pro3-2
```

```python
import numpy as np
from pydrake.all import (MathematicalProgram, Solve)

A = np.mat([[0, 1, 0], [0, 0, 1], [-1, -3, -3]])

b = np.array([[0], [0], [1]])
c = np.array([[1], [0], [0]])
alpha = 0.7
beta = 0.1

# Step 2: formulate the S.D.P.
prog = MathematicalProgram()
num_states = 3
P = prog.NewSymmetricContinuousVariables(num_states, "P")
tau = prog.NewContinuousVariables(1, "tau")

I = np.identity(3)
G_1row =np.hstack((np.array(-(A.T@P+P@A+beta*P)+tau*alpha**2*c@c.T)
                   ,np.array(-P@b)))
G_2row = np.hstack((np.array(-b.T@P),np.array([-tau])))
G = np.vstack((G_1row,G_2row))
# print(G.shape)

# find P,tau
#prog.AddLinearCost(np.trace(X))

# s.t.
prog.AddPositiveSemidefiniteConstraint(P - .01 *np.identity(num_states))

prog.AddPositiveSemidefiniteConstraint(G)

result = Solve(prog)

if result.is_success():
    P = result.GetSolution(P)
    tau = result.GetSolution(tau)
    print("Succeed to find the solution! ")
    print("P = \n", P)
    print("tau = \n ", tau)
else:
    print("no result")
```

```
Succeed to find the solution!
P =
 [[44.86675378 36.00366958 12.571446   ]
 [36.00366958 89.66902601 21.46918172]
 [12.571446   21.46918172 19.0368192 ]]
tau =
  [-21.89290121]
```

Problem 4

Stabilization via LMIs: Consider the time-varying LDS (linear dynamical system)

$$\dot{x}(t) = A(t)x(t) + Bu(t),$$

with $x(t) \in R^n$ and $u(t) \in R^m$, where $A(t) \in A_1, \ldots, A_M$. Thus, the dynamics matrix $A(t)$ can take any of M values, at any time. We seek a linear state feedback gain matrix K ∊ Rm×n for which the closed-loop system

$$\dot{x}(t) = [A(t) + BK]x(t),$$

is globally asymptotically stable. But even if you are given a specific state feedback gain matrix K, this is very hard to determine. So we will require the existence of a quadratic Lyapunov function that establishes exponential stability of the closed-loop system, i.e., a matrix $P = PT \succ 0$ for which

$$\dot{V}(z,t) = z^T[(A(t)+BK)^T P + P(A(t)+BK)]z \leq -\beta V(z)$$

for all z, and for any possible value of $A(t)$. (The parameter $\beta > 0$ is given, and sets a minimum decay rate for the closed-loop trajectories.) So roughly speaking we seek

- a stabilizing state feedback gain, and
- a quadratic Lyapunov function that certifies the closed-loop performance.

In this problem, you will use LMIs to find both K and P, simultaneously.

(a) Pose the problem of finding P and K as an LMI problem. Hint: Starting from the inequality above, you will not get an LMI in the variables P and K (although you will have a set of matrix inequalities that are affine in K, for fixed P, and linear in P, for fixed K). Use the new variables $X = P^{-1}$ and $Y = KP^{-1}$. Be sure to explain why you can change variables.

(b) Carry out your method for the specific problem instance

$$A_1 = \begin{bmatrix} -0.5 & 0.3 & 0.4 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad A_2 = \begin{bmatrix} -0.7 & 0.1 & -0.2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 0.6 & -0.7 & 0.2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$B = (1, 0, 0)$, and $\beta = 1$. (Thus, we require a closed-loop decay at least as fast as $e^{-t/2}$.)

Solution

(a) For the time varying linear system, asymptotically stable is equivalent to exponential stable. For the Lyapunov function $V(z,t) = z^T P z$, the condition for stable is

$$P > 0$$
$$\dot{V}(z,t) = z^T[(A(t)+BK)^T P + P(A(t)+BK)]z \leq -\beta V(z)$$

For the second conditon we can get

$$z^T[(A(t)+BK)^T P + P(A(t)+BK) + \beta P]z \leq 0$$

Define $X = P^{-1}, Y = KP^{-1}$, P is symmetric matrix $P = P^T$, multiply P^{-1} to the both sides of the inequality, then the above inequality can be

$$\begin{aligned} XA(t)^T + Y^T B^T + A(t)X + BY + \beta X \leq 0 \\ A(t)X + (A(t)X)^T + BY + (BY)^T + \beta X \leq 0 \\ -(A(t)X + (A(t)X)^T + BY + (BY)^T + \beta X) \geq 0. \end{aligned}$$

In order to get a minimum decay rate for the clossed-loop trajectories, we must get β maximum, so the problem can be formulated by

$$
\begin{cases}
\min_{X,Y} & -\beta \\
\text{subject.:} & X^{-1} \succ 0, \\
 & -(A(t)X + (A(t)X)^T + BY + (BY)^T + \beta X) \succeq 0
\end{cases}
$$

In [ ]:

```python
# pro4-2
import numpy as np
from pydrake.all import (MathematicalProgram, Solve)

# Step 1: define dynamics matrices A_i(t) and corresponding parameters
A = []
A.append(np.mat([[-0.5, 0.3, 0.4], [1, 0, 0], [0, 1, 0]]))
A.append(np.mat([[-0.7, 0.1, -0.2], [1, 0, 0], [0, 1, 0]]))
A.append(np.mat([[0.6, -0.7, 0.2], [1, 0, 0], [0, 1, 0]]))

# B = np.mat([[1], [0], [0]])
B = np.array([1, 0, 0])
beta = 1.0

# Step 2: formulate the S.D.P.
prog = MathematicalProgram()
num_states = 3
X = prog.NewSymmetricContinuousVariables(num_states, "X")
Y = prog.NewContinuousVariables(num_states, "Y")

# find X,Y
#prog.AddLinearCost(np.trace(X))

# s.t.
prog.AddPositiveSemidefiniteConstraint(X - .001 * np.identity(num_states))
for i in range(len(A)):
    prog.AddPositiveSemidefiniteConstraint(
        -(X.dot(A[i].transpose()) + Y.transpose().dot(B.transpose()) +
          A[i].dot(X) + B.dot(Y) + beta * X - .1 * np.identity(num_states))
    )

result = Solve(prog)
if result.is_success():
    X = result.GetSolution(X)
    Y = result.GetSolution(Y)
    P = np.linalg.inv(X)
    K = Y.dot(P)
    print("Succeed to find the solution! ")
    print("P = \n", P)
    print("K = \n ", K)
else:
    print("no result")
```

```
Succeed to find the solution!
P =
 [[ 0.2244294  -0.14487089 -0.19088273]
 [-0.14487089  0.13899062  0.09789311]
 [-0.19088273  0.09789311  0.19386985]]
K =
  [-21.97083237  14.18233962  18.68673382]
```