Controle robusto e relaxações de LMI

Guilherme Coelho Bogado

Resumo **- Este relatório refere-se ao estudo orientado sobre Controle robusto e relaxações de LMI, no qual é introduzido o conceito de controle robusto e o papel fundamental de relaxações de LMI na solução de problemas involvendo incertezas. Os resultados foram obtidos a partir de simulações usando do solver Mosek e usando a biblioteca CVXPY em Python. Também, usou-se do scilab para verificar os valores obitdos.**

I. Introdução

EM controle robusto, é estudado a performance de um sistema de controle sobre a mudança de seus parâmetros, denominada incerteza. A abordagem de tal sistemas é projetar compensadores que operam sobre incertezas, no qual é idealizado para acomodar uma faixa dentro o parâmetro incerto, garantindo desempenho. Uma das maneiras de modelar a variação dos parâmetros do sistema é a partir de um modelo matemático para a incerteza, sendo dois modelos, a incerteza politópica e incerteza limitada em norma.

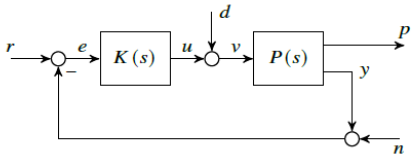

Fig. 1: Planta com incerteza introduzida

Um sistema incerto pode ser descrito como:

$$\begin{cases} \dot{x} = A(\alpha)x + B(\alpha)u \\ y = C(\alpha)x + D(\alpha)u \end{cases} \tag{1}$$

No qual α é um vetor de incertezas, porém fixo no tempo. Aplicando Lyapunov e fazendo $u = 0$, obtemos:

$$\begin{aligned} V(x) &= x^{'}Px, P \succ 0 \\ \dot{V}(x) &= x^{'}P\dot{x} + \dot{x}^{'}Px \end{aligned} \tag{2}$$

Substituindo $\dot{x}$ por $A(\alpha)x$:

$$x^{'}PA(\alpha)x + x^{'}[A(\alpha)^{'}]Px \prec 0 \tag{3}$$

Simplificando, obtemos:

$$x^{'}[PA(\alpha) + (A(\alpha))^{'}P]x \prec 0 \tag{4}$$

Considerando apenas os colchetes:

$$PA(\alpha) + (A(\alpha))^{'}P \prec 0 \tag{5}$$

A partir das equações 4 e 5 podemos modelar as incertezas.

A. Incerteza politópica

Incerteza politópica é um tipo de incerteza delimitada por um politopo, um conjunto convexo definido como a combinação convexa de vértices, considere que a matriz A varia dentro de um conjunto politópico $A \in conv(A_1, A_2, ..., A_n)$, qualquer dessas matrizes podem ser expressas como:

$$A = \sum_{i=1}^{n} \alpha_i A_i \quad \text{com} \quad \sum_{i=1}^{n} \alpha_i = 1 \quad \text{e} \quad \alpha_i \geq 0 \tag{6}$$

Para cada incerteza α em $A \in \mathbb{R}^{nxn}$, o número de extremos pode chegar a 2^{n^2}

Considerando o fato de que a estabilidade dos vértices não garantem a estabilidade do politopo, sendo apenas condição necessária e que se $A(\alpha)$ é estável, então para qualquer $x > 0, xA(\alpha)$ é estável, uma condição necessária e suficiente para garantir a estabiliadade robusta de $A(\alpha)$ é dada a por $P(\alpha) = P'(\alpha) \prec 0$ e a partir da equação 4 tal que:

$$A(\alpha)'P(\alpha) + P(\alpha)A(\alpha) \prec 0 \tag{7}$$

Com isso, se existir uma matriz P que satisfaça 7, então o sistema é estável para todos $A(\alpha)$.

II. Procedimento Experimental

Para exemplificar a teoria apresentada na introdução, fez-se exemplos de LMIs em python, no qual usou-se das bibliotecas CVXPY e NumPy e do solver Mosek.

A. Incerteza Politópica

Na simulação, considerou-se a matriz $A \in \mathbb{R}^{3x3}$ com incertezas em a_{11} e a_{23}, com $a_{11} \in [1, 1.5]$ e $a_{23} \in [0.1, 0.5]$, com isso, tem 4 matrizes A possíveis para a LMI, também considerou a matriz B como $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$, para obter a matriz W, da LMI $AP + PA^{'} + WB^{'} + BW^{'} \prec 0$, que encontra o compensador $K = P^{-1}W$

```
import cvxpy as cp
import numpy as np
import matplotlib.pyplot as plt


a11_max = 1.5
a11_min = 1.0
a23_max = 0.5
a23_min = 0.1

A1 = np.array([[a11_min, 0.2, 0.1],
               [0.0, -1.0, a23_min],
```

```
                [0.0, 0.0, -0.5]])

A2 = np.array([[a11_max, 0.2, 0.1],
               [0.0, -1.0, a23_min],
               [0.0, 0.0, -0.5]])

A3 = np.array([[a11_min, 0.2, 0.1],
               [0.0, -1.0, a23_max],
               [0.0, 0.5, -0.5]])

A4 = np.array([[a11_max, 0.2, 0.1],
               [0.0, -1.0, a23_max],
               [0.0, 0.5, -0.5]])

matrizes = [A1, A2, A3, A4]

B =np.array([[0.0],
             [0.0],
             [1.0]])


P = cp.Variable((3, 3), symmetric=True)
W = cp.Variable((3, 1))



constraints = [P - np.eye(3) >>  0]



constraints_W = []
for A in matrizes:
    constraints_W.append(A @ P + P @ A.T + W
    @ B.T + B @ W.T << - np.eye(3) )


constraints += constraints_W



prob_W = cp.Problem(cp.Minimize(0),
    constraints)
prob_W.solve(solver=cp.MOSEK)
autovalorP = np.linalg.eigvals(P.value)
Wv = W.value
Pv = P.value
K = Wv.T @ np.linalg.inv(Pv)
print("Matriz P: \n")
print(P.value)
print(" ")
print("Matriz W:\n")
print(W.value)
print(" ")
print("Valor K:")
print(K)
print(" ")
print("Autovalores P:")
print(autovalorP)
```

O código, tem como input os parâmetros de a_{11} e a_{23} e gera a saída da matriz P e W, assim, para as incertezas citadas acima, obteve-se a matriz P:

```
[[   44.081197    -204.31206187
   -444.56994717]
 [ -204.31206187  2460.13698065
   -119.69379509]
 [ -444.56994717  -119.69379509
   11837.77441755]]
```

E a matriz W para encontrar o compensador K:

```
[[ -654.88049589]
[-4313.88110306]
[ 2803.9964785 ]]
```

III. Análise dos Dados

Para verificar os dados obtidos do programa do procedimento experimental, usou-se do Scilab, então, testou-se os valores dos autovalores de P usando o comando spec, obtendo os autovalores de P: $\begin{bmatrix} 1.185584 \\ 9.57037 \\ 2.476579 \end{bmatrix}$ Com isso, garante-se que P é definida positiva. Com os valores de W e P, podemos calcular K, obtendo: $\begin{bmatrix} -94.62545438 \\ -9.77825158 \\ -3.41567861 \end{bmatrix}$ Também, verificou-se os autovalores das LMIs $A_i^T P + P A_i \prec 0$ testadas, obtendo:

$$A_1 = \begin{bmatrix} -2.15471061 \\ -0.00901439 \\ -0.68629874 \end{bmatrix} \; A_2 = \begin{bmatrix} -1.39588432 \\ -2.1497322 \\ -6.88069 \end{bmatrix} \tag{8}$$

$$A_3 = \begin{bmatrix} -0.00902 \\ -0.70159 \\ -2.208230 \end{bmatrix} \; A_3 = \begin{bmatrix} -0.006262 \\ -2.208737 \\ -0.693016 \end{bmatrix} \tag{9}$$

Com isso, conclui-se que as LMIs possuem autovalores negativos, logo, o sistema é robustamente estável. Para verificar graficamente a estabilidade do sistema, foi simulado considerando a váriavel C como $C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$, entrada degrau e condição inicial $x0$ como $x0 = \begin{bmatrix} 1 \\ -0.5 \\ 0.2 \end{bmatrix}$ e foi usado os comandos $syslin$ para definir o SLIT-C , $csim$ para simula-lo e $plot$ para plotar o resultado. Assim, foi obtido o gráfico:

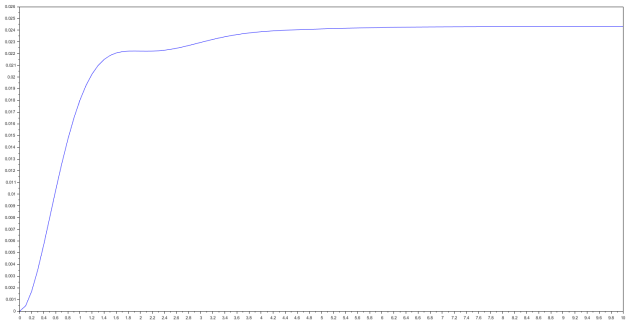

Fig. 2: Gráfico de estabilidade

Assim, observando o gráfico, percebe-se que o sistema é estável, com um overshoot e depois converge a um valor positivo.

IV. Referências Bibliográficas

[1] [2]

[1] K. Ogata *et al.*, *Modern control engineering*. Prentice Hall India, 2009.
[2] P. L. D. Peres, "Análise e controle de sistemas lineares por desigualdades matriciais lineares (lmis)," https://www.fee.unicamp.br/profs/ricfow/IA892/ia892.htm, accessed: 2010-09-30.