UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO DEPARTAMENTE DE SISTEMAS E ENERGIA

# **Giovanni Chemello Caprio**

# **RA: 211483**

**TRABALHO COMPUTACIONAL**

**SEGUNDO SEMESTRE 2018**

1. **Resultados**

Os Lemas e Teoremas foram aplicados para os seguintes casos:

* Escalar variando entre (10-5, 10-3, 10-1, 1);
* Escalar igual a um para todos os estágios;
* Na criação do ganho no primeiro estágio, será gerado de forma politópica. Ou seja, Z terá vértices. Sabemos que a matriz C do controlador será politópica, sendo impossível a implementação, mas serão testadas para analise de possíveis melhoras nos resultados.

Para o caso do escalar variando, o código foi gerado e apresentou os seguintes resultados no Matlab:

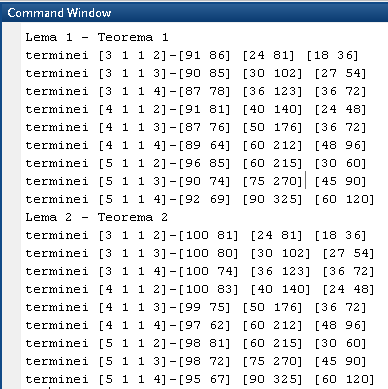


Figure 1 - Resultados Matlab (Escalar Variável)

Colocadondo os resultados em forma de tabela para uma melhor verificação sendo SE o número de sistemas estabilizados, V o número de variáveis escalares e L o número de linhas de LMIs, para cada valor do escalar:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | | |
|  | **L1** | | | **T1** | | | **L2** | | | **T2** | | |
| *(n,m,p,N)* | *V* | *L* | SE | *V* | *L* | SE | *V* | *L* | SE | *V* | *L* | SE |
| (3,1,1,2) | 24 | 18 | 91 | 81 | 36 | 86 | 24 | 18 | 100 | 81 | 36 | 81 |
| (3,1,1,3) | 30 | 27 | 90 | 102 | 54 | 85 | 30 | 27 | 100 | 102 | 54 | 80 |
| (3,1,1,4) | 36 | 36 | 87 | 123 | 72 | 78 | 36 | 36 | 100 | 123 | 72 | 74 |
| (4,1,1,2) | 40 | 24 | 91 | 140 | 48 | 81 | 40 | 24 | 100 | 140 | 48 | 83 |
| (4,1,1,3) | 50 | 36 | 87 | 176 | 72 | 76 | 50 | 36 | 99 | 176 | 72 | 75 |
| (4,1,1,4) | 60 | 48 | 89 | 212 | 96 | 64 | 60 | 48 | 97 | 212 | 96 | 62 |
| (5,1,1,2) | 60 | 30 | 96 | 215 | 60 | 85 | 60 | 30 | 98 | 215 | 60 | 81 |
| (5,1,1,3) | 75 | 45 | 90 | 270 | 90 | 74 | 75 | 45 | 98 | 270 | 90 | 72 |
| (5,1,1,4) | 90 | 60 | 92 | 325 | 120 | 69 | 90 | 60 | 95 | 325 | 120 | 67 |
|  |  |  | 90.3 |  |  | 77.5 |  |  | 98.5 |  |  | 75 |
|  |  |  |  |  |  | 85.8 |  |  |  |  |  | 76.1 |
|  | | | | | | | | | | | | |

Para o caso do escalar fixo em um, o código similar ao primeiro caso foi gerado e apresentou os seguintes resultados no Matlab:

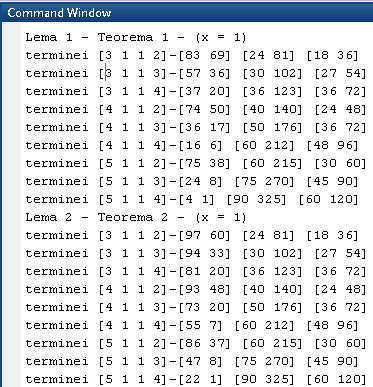


Figure 2 Resultados Matlab (Escalar Fixo)

Para facilitar as analises e organizar de uma maneira mais visual, uma tabela similar a anterior foi gerada:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | | |
|  | **L1** | | | **T1** | | | **L2** | | | **T2** | | |
| *(n,m,p,N)* | *V* | *L* | SE | *V* | *L* | SE | *V* | *L* | SE | *V* | *L* | SE |
| (3,1,1,2) | 24 | 18 | 83 | 81 | 36 | 69 | 24 | 18 | 97 | 81 | 36 | 60 |
| (3,1,1,3) | 30 | 27 | 57 | 102 | 54 | 36 | 30 | 27 | 94 | 102 | 54 | 33 |
| (3,1,1,4) | 36 | 36 | 37 | 123 | 72 | 20 | 36 | 36 | 81 | 123 | 72 | 20 |
| (4,1,1,2) | 40 | 24 | 74 | 140 | 48 | 50 | 40 | 24 | 93 | 140 | 48 | 48 |
| (4,1,1,3) | 50 | 36 | 36 | 176 | 72 | 17 | 50 | 36 | 73 | 176 | 72 | 20 |
| (4,1,1,4) | 60 | 48 | 16 | 212 | 96 | 6 | 60 | 48 | 55 | 212 | 96 | 7 |
| (5,1,1,2) | 60 | 30 | 75 | 215 | 60 | 38 | 60 | 30 | 86 | 215 | 60 | 37 |
| (5,1,1,3) | 75 | 45 | 24 | 270 | 90 | 8 | 75 | 45 | 47 | 270 | 90 | 8 |
| (5,1,1,4) | 90 | 60 | 4 | 325 | 120 | 1 | 90 | 60 | 22 | 325 | 120 | 1 |
|  |  |  | 45.1 |  |  | 27.2 |  |  | 72 |  |  | 26 |
|  |  |  |  |  |  | 60.3 |  |  |  |  |  | 36.1 |
|  | | | | | | | | | | | | |

# 

Para finalizar, foi gerado um programa onde o Z seria em função de alpha, ou seja, tendo vértices e gerando um ganho K com vértices, para o primeiro estágio (Lema 1). Apresentando resultados de estabilização, tanto para o Lema 1 como para o Teorema 1, como podemos ver a seguir pela solução obtida no Matlab:

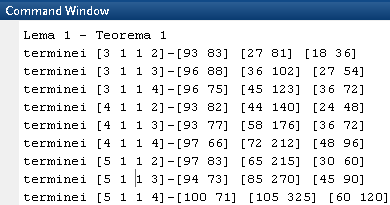


Figure 3 Resultados Matlab ( Z(a) )

Assim como nos exemplos anteriores, uma tabela similar foi gerada para analises:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **L1** | | | **T1** | | |
| *(n,m,p,N)* | *V* | *L* | SE | *V* | *L* | SE |
| (3,1,1,2) | 27 | 18 | 93 | 81 | 36 | 83 |
| (3,1,1,3) | 36 | 27 | 96 | 102 | 54 | 88 |
| (3,1,1,4) | 45 | 36 | 96 | 123 | 72 | 75 |
| (4,1,1,2) | 44 | 24 | 93 | 140 | 48 | 82 |
| (4,1,1,3) | 58 | 36 | 93 | 176 | 72 | 77 |
| (4,1,1,4) | 72 | 48 | 97 | 212 | 96 | 66 |
| (5,1,1,2) | 65 | 30 | 97 | 215 | 60 | 83 |
| (5,1,1,3) | 85 | 45 | 94 | 270 | 90 | 73 |
| (5,1,1,4) | 105 | 60 | 100 | 325 | 120 | 71 |
|  |  |  | 95.4 |  |  | 77.5 |
|  |  |  |  |  |  | 81.23 |
|  |  |  |  |  |  |  |

1. **Conclusões**

Inicialmente, os resultados pareceram ser bem satisfatórios. Provavelmente, uma maior busca nos escalares geraria resultados melhores. Se percebe isso pelo teste do escalar sendo um, apenas no Lema 2 a perda de sistemas estabilizados não foi grande, em todos outros teve perdas de aproximadamente 50%, retirando a busca em três escalares (10-5, 10-3, 10-1). Entretanto, neste mesmo exemplo foi claro a diferença de custo computacional para a busca entre menos escalares, quanto menos escalares, menor é seu custo computacional.

Para a proposta de ter um Z dependente de alpha, obtivemos resultados interessantes. No Lema 1, obtivemos 5,1% de melhora de sistemas estabilizados. Entretanto, para o Teorema 1 obtivemos uma piora de 4,6% dos sistemas estabilizados, para caso o L1 tenha obtido êxito na estabilização. Além de terem sido gerados mais variáveis para o L1, obviamente. Ou seja, não houve uma melhora tão clara nos resultados obtidos para um Z dependente, para os testes realizados.

1. **Apêndices**

* **Código Principal**

%% Trabalho Final - IA 892

clc, clear all, close all

%% Inicialização

xs = [10^(-5) 10^(-3) 10^(-1) 1];

load('DB\_dof.mat');

output.tabela1 = [];

output.tabela2 = [];

%% Lema 1 - Teorema 1

display('Lema 1 - Teorema 1')

for d=1:size(dimensoes,1)

ordem = dimensoes(d,1);

entradas = dimensoes(d,2);

saidas = dimensoes(d,3);

vertices = dimensoes(d,4);

placar1 = [0 0]; % Soma Estaveis

placar\_v = [0 0]; % Variáveis

placar\_l = [0 0]; % Linhas de LMI

for i= 1:totalSistemas

A = BASE{ordem,entradas,saidas,vertices,i}.A;

B = BASE{ordem,entradas,saidas,vertices,i}.B;

C = BASE{ordem,entradas,saidas,vertices,i}.C;

feas = [0 0];

feas\_1 = 0;

feas\_2 = 0;

for t = 1:4

x = xs(t);

[feas\_1,K,L1,V1] = primeiroEstagioK(A,B,x,vertices); % 1 Lema

placar\_v(1) = V1;

placar\_l(1) = L1;

if feas\_1 == 1

feas = [1 0];

[feas\_2,Ac,Bc,L2,V2] = segundoEstagio\_Cc(A,B,C,K,x,vertices); % 1 Teorema

placar\_v(2) = V2;

placar\_l(2) = L2;

if feas\_2 == 1

feas = [1 1];

break;

end

end

end

placar1 = placar1 + feas;

end

% Printar na tela e armazenar iteração

fprintf('terminei [%d %d %d %d]-[%d %d] [%d %d] [%d %d] \n',ordem,entradas,saidas,vertices...

,placar1(1),placar1(2),placar\_v(1),placar\_v(2),placar\_l(1),placar\_l(2));

output.tabela1 = [output.tabela1; [ordem,entradas,saidas,vertices,placar1(1),placar1(2)...

,placar\_v(1),placar\_v(2),placar\_l(1),placar\_l(2)]];

end

%% Lema 2 - Teorema 2

display('Lema 2 - Teorema 2')

for d=1:size(dimensoes,1)

ordem = dimensoes(d,1);

entradas = dimensoes(d,2);

saidas = dimensoes(d,3);

vertices = dimensoes(d,4);

placar2 = [0 0]; % Soma Estaveis

placar\_v = [0 0]; % Variáveis

placar\_l = [0 0]; % Linhas de LMI

for i= 1:totalSistemas

A = BASE{ordem,entradas,saidas,vertices,i}.A;

B = BASE{ordem,entradas,saidas,vertices,i}.B;

C = BASE{ordem,entradas,saidas,vertices,i}.C;

feas = [0 0];

feas\_1 = 0;

feas\_2 = 0;

for t = 1:4

x = xs(t);

[feas\_1,L,L1,V1] = primeiroEstagioL(A,C,x,vertices); % 2 Lema

placar\_v(1) = V1;

placar\_l(1) = L1;

if feas\_1 == 1

feas = [1 0];

[feas\_2,Ac,Cc,L2,V2] = segundoEstagio\_Bc(A,B,C,L,x,vertices); % 2 Teorema

placar\_v(2) = V2;

placar\_l(2) = L2;

if feas\_2 == 1

feas = [1 1];

break;

end

end

end

placar2 = placar2 + feas;

end

% Printar na tela e armazenar iteração

fprintf('terminei [%d %d %d %d]-[%d %d] [%d %d] [%d %d] \n',ordem,entradas,saidas,vertices...

,placar2(1),placar2(2),placar\_v(1),placar\_v(2),placar\_l(1),placar\_l(2));

output.tabela2 = [output.tabela2; [ordem,entradas,saidas,vertices,placar2(1),placar2(2)...

,placar\_v(1),placar\_v(2),placar\_l(1),placar\_l(2)]];

end

* **Código (escalar = 1)**

%% Trabalho Final - IA 892

clc, clear all, close all

%% Inicialização

%xs = [10^(-5) 10^(-3) 10^(-1) 1];

xs = 1;

load('DB\_dof.mat');

output.tabela1 = [];

output.tabela2 = [];

%% Lema 1 - Teorema 1

display('Lema 1 - Teorema 1 - (x = 1)')

for d=1:size(dimensoes,1)

ordem = dimensoes(d,1);

entradas = dimensoes(d,2);

saidas = dimensoes(d,3);

vertices = dimensoes(d,4);

placar1 = [0 0]; % Soma Estaveis

placar\_v = [0 0]; % Variáveis

placar\_l = [0 0]; % Linhas de LMI

for i= 1:totalSistemas

A = BASE{ordem,entradas,saidas,vertices,i}.A;

B = BASE{ordem,entradas,saidas,vertices,i}.B;

C = BASE{ordem,entradas,saidas,vertices,i}.C;

feas = [0 0];

feas\_1 = 0;

feas\_2 = 0;

for t = 1:1

x = xs;

[feas\_1,K,L1,V1] = primeiroEstagioK(A,B,x,vertices); % 1 Lema

placar\_v(1) = V1;

placar\_l(1) = L1;

if feas\_1 == 1

feas = [1 0];

[feas\_2,Ac,Bc,L2,V2] = segundoEstagio\_Cc(A,B,C,K,x,vertices); % 1 Teorema

placar\_v(2) = V2;

placar\_l(2) = L2;

if feas\_2 == 1

feas = [1 1];

break;

end

end

end

placar1 = placar1 + feas;

end

% Printar na tela e armazenar iteração

fprintf('terminei [%d %d %d %d]-[%d %d] [%d %d] [%d %d] \n',ordem,entradas,saidas,vertices...

,placar1(1),placar1(2),placar\_v(1),placar\_v(2),placar\_l(1),placar\_l(2));

output.tabela1 = [output.tabela1; [ordem,entradas,saidas,vertices,placar1(1),placar1(2)...

,placar\_v(1),placar\_v(2),placar\_l(1),placar\_l(2)]];

end

%% Lema 2 - Teorema 2

display('Lema 2 - Teorema 2 - (x = 1)')

for d=1:size(dimensoes,1)

ordem = dimensoes(d,1);

entradas = dimensoes(d,2);

saidas = dimensoes(d,3);

vertices = dimensoes(d,4);

placar2 = [0 0]; % Soma Estaveis

placar\_v = [0 0]; % Variáveis

placar\_l = [0 0]; % Linhas de LMI

for i= 1:totalSistemas

A = BASE{ordem,entradas,saidas,vertices,i}.A;

B = BASE{ordem,entradas,saidas,vertices,i}.B;

C = BASE{ordem,entradas,saidas,vertices,i}.C;

feas = [0 0];

feas\_1 = 0;

feas\_2 = 0;

for t = 1:1

x = xs;

[feas\_1,L,L1,V1] = primeiroEstagioL(A,C,x,vertices); % 2 Lema

placar\_v(1) = V1;

placar\_l(1) = L1;

if feas\_1 == 1

feas = [1 0];

[feas\_2,Ac,Cc,L2,V2] = segundoEstagio\_Bc(A,B,C,L,x,vertices); % 2 Teorema

placar\_v(2) = V2;

placar\_l(2) = L2;

if feas\_2 == 1

feas = [1 1];

break;

end

end

end

placar2 = placar2 + feas;

end

% Printar na tela e armazenar iteração

fprintf('terminei [%d %d %d %d]-[%d %d] [%d %d] [%d %d] \n',ordem,entradas,saidas,vertices...

,placar2(1),placar2(2),placar\_v(1),placar\_v(2),placar\_l(1),placar\_l(2));

output.tabela2 = [output.tabela2; [ordem,entradas,saidas,vertices,placar2(1),placar2(2)...

,placar\_v(1),placar\_v(2),placar\_l(1),placar\_l(2)]];

end

* **Funçao K (1º Estágio)**

function [feas,K,linhas,V] = primeiroEstagioK(A,B,x,vertices)

% Inicialização

linhas = 0; % LMI contagem de linhas

n = size(A{1},1);

Ai= [];

Bi= [];

LMIs = [];

for i = 1:vertices

Ai = [Ai A{i}];

Bi = [Bi B{i}];

end

%Criação das Variáveis

polyA = rolmipvar(Ai,'A(a)',vertices,1);

polyB = rolmipvar(Bi,'B(a)',vertices,1);

polyP = rolmipvar(n,n,'P(a)','symmetric',vertices,1);

Z = rolmipvar(1,n,'Z','full',vertices,0);

X = rolmipvar(n,n,'X','full',vertices,0);

% LMIs

LMIs = [LMIs, polyP >= 0];

linhas = linhas + n; % Soma grau de P

T11 = polyA\*X + X'\*polyA' + polyB\*Z + Z'\*polyB';

T12 = polyP - X' + x\*polyA\*X + x\*polyB\*Z;

T22 = -x\*(X + X');

T = [ T11 T12;

T12' T22];

LMIs = [LMIs, T <= 0];

linhas = linhas + 2\*n; % Soma grau de T

% Resolução

optimize(LMIs,[],sdpsettings('verbose',0,'solver','sedumi'));

res = min(checkset(LMIs));

% Número de Variáveis e Linhas

V = size(getvariables(LMIs),2);

linhas = vertices\*linhas; % LMIs totais

if res > 0

feas = 1;

K = double(Z)\*inv(double(X));

else

feas = 0;

K = NaN;

end

end

* **Função L (1º Estágio)**

function [feas,L,linhas,V] = primeiroEstagioL(A,C,x,v)

% Inicialização

linhas = 0; % LMI contagem de linhas

n = size(A{1},1);

Ai= [];

Ci= [];

LMIs = [];

for i = 1:v

Ai = [Ai A{i}];

Ci = [Ci C{i}];

end

%Criação das Variáveis

polyA = rolmipvar(Ai,'A(a)',v,1);

polyC = rolmipvar(Ci,'C(a)',v,1);

polyP = rolmipvar(n,n,'P(a)','symmetric',v,1);

Z = rolmipvar(n,1,'Z','full',v,0);

X = rolmipvar(n,n,'X','full',v,0);

% LMIs

LMIs = [LMIs, polyP >= 0];

linhas = linhas + n; % Soma grau de P

T11 = X\*polyA + polyA'\*X' + Z\*polyC + polyC'\*Z';

T12 = polyP - X + x\*polyA'\*X' + x\*polyC'\*Z';

T22 = -x\*(X + X');

T = [ T11 T12;

T12' T22];

LMIs = [LMIs, T <= 0];

linhas = linhas + 2\*n; % Soma grau de T

% Resolução

optimize(LMIs,[],sdpsettings('verbose',0,'solver','sedumi'));

res = min(checkset(LMIs));

% Número de Variáveis

V = size(getvariables(LMIs),2);

linhas = v\*linhas;

if res > 0

feas = 1;

L = inv(double(X))\*double(Z);

else

feas = 0;

L = NaN;

end

end

* **Função Cc (2º Estágio)**

function [feas\_2,Ac,Bc,linhas,V] = segundoEstagio\_Cc(A,B,C,K,x,vertices)

% Inicialização

linhas = 0; % LMI contagem de linhas

n = size(A{1},1);

p = size(C{1},1);

Ai = [];

Bi = [];

Ci = [];

LMIs = [];

for i = 1:vertices

Ai = [Ai A{i}];

Bi = [Bi B{i}];

Ci = [Ci C{i}];

end

%Criação das Variáveis

polyA = rolmipvar(Ai,'A(a)',vertices,1);

polyB = rolmipvar(Bi,'B(a)',vertices,1);

polyC = rolmipvar(Ci,'C(a)',vertices,1);

polyP = rolmipvar(2\*n,2\*n,'P(a)','symmetric',vertices,1);

G = rolmipvar(n,p,'G','full',vertices,0);

V = rolmipvar(n,n,'V','full',vertices,0);

H = rolmipvar(n,n,'H','full',vertices,0);

Q = rolmipvar(n,n,'Q','full',vertices,0);

Y = rolmipvar(n,n,'Y','full',vertices,0);

% LMIs

LMIs = [LMIs, polyP >= 0];

linhas = linhas + 2\*n; % Soma grau de P

J = [ Q Q ;

(Y+V) Y ];

T11 = Q\*(polyA + polyB\*K);

T12 = Q\* polyA;

T21 = Y\*(polyA + polyB\*K) + G\*polyC + H;

T22 = Y\*polyA+G\*polyC;

T = [ T11 T12;

T21 T22];

F11 = T + T';

F12 = polyP - J' + x\*T;

F22 = -x\*(J + J');

F = [ F11 F12;

F12' F22];

LMIs = [LMIs, F <= 0];

linhas = linhas + 4\*n; % Soma grau de F

% Resolução

optimize(LMIs,[],sdpsettings('verbose',0,'solver','sedumi'));

res = min(checkset(LMIs));

% Número de Variáveis e Linhas

V = size(getvariables(LMIs),2);

linhas = vertices\*linhas; % LMIs totais

if res > 0

feas\_2 = 1;

Ac = inv(double(V))\*double(H);

Bc = inv(double(V))\*double(G);

else

feas\_2 = 0;

Ac = NaN;

Bc = NaN;

end

end

* **Função Bc (2º Estágio)**

function [feas\_2,Ac,Cc,linhas,V] = segundoEstagio\_Bc(A,B,C,L,x,vertices)

% Inicialização

linhas = 0; % LMI contagem de linhas

n = size(A{1},1);

m = size(B(1),1);

Ai = [];

Bi = [];

Ci = [];

LMIs = [];

for i = 1:vertices

Ai = [Ai A{i}];

Bi = [Bi B{i}];

Ci = [Ci C{i}];

end

%Criação das Variáveis

polyA = rolmipvar(Ai,'A(a)',vertices,1);

polyB = rolmipvar(Bi,'B(a)',vertices,1);

polyC = rolmipvar(Ci,'C(a)',vertices,1);

polyP = rolmipvar(2\*n,2\*n,'P(a)','symmetric',vertices,1);

G = rolmipvar(n,m,'G','full',vertices,0);

V = rolmipvar(n,n,'V','full',vertices,0);

H = rolmipvar(n,n,'H','full',vertices,0);

Q = rolmipvar(n,n,'Q','full',vertices,0);

Y = rolmipvar(n,n,'Y','full',vertices,0);

% LMIs

LMIs = [LMIs, polyP >= 0];

linhas = linhas + 2\*n; % Soma grau de P

J = [ Q Q ;

(Y+V) Y ];

T11 = Q\*(polyA' + polyC'\*L');

T12 = Q\* polyA';

T21 = Y\*(polyA' + polyC'\*L') + G\*polyB' + H;

T22 = Y\*polyA'+G\*polyB';

T = [ T11 T12;

T21 T22];

F11 = T + T';

F12 = polyP - J' + x\*T;

F22 = -x\*(J + J');

F = [ F11 F12;

F12' F22];

LMIs = [LMIs, F <= 0];

linhas = linhas + 4\*n; % Soma grau de F

% Resolução

optimize(LMIs,[],sdpsettings('verbose',0,'solver','sedumi'));

res = min(checkset(LMIs));

% Número de Variáveis

V = size(getvariables(LMIs),2);

linhas = vertices\*linhas; % LMIs totais

if res > 0

feas\_2 = 1;

Ac = inv(double(V))\*double(H);

Cc = inv(double(V))\*double(G);

else

feas\_2 = 0;

Ac = NaN;

Cc = NaN;

end

end

* **Extra ( Z(a) )**

function [feas,K,linhas,V] = primeiroEstagioK(A,B,x,vertices)

% Inicialização

linhas = 0; % LMI contagem de linhas

n = size(A{1},1);

Ai= [];

Bi= [];

LMIs = [];

for i = 1:vertices

Ai = [Ai A{i}];

Bi = [Bi B{i}];

end

%Criação das Variáveis

polyA = rolmipvar(Ai,'A(a)',vertices,1);

polyB = rolmipvar(Bi,'B(a)',vertices,1);

polyP = rolmipvar(n,n,'P(a)','symmetric',vertices,1);

Z = rolmipvar(1,n,'Z(a)','full',vertices,1); % TORNA O Z em Z(a)

X = rolmipvar(n,n,'X','full',vertices,0);

% LMIs

LMIs = [LMIs, polyP >= 0];

linhas = linhas + n; % Soma grau de P

T11 = polyA\*X + X'\*polyA' + polyB\*Z + Z'\*polyB';

T12 = polyP - X' + x\*polyA\*X + x\*polyB\*Z;

T22 = -x\*(X + X');

T = [ T11 T12;

T12' T22];

LMIs = [LMIs, T <= 0];

linhas = linhas + 2\*n; % Soma grau de T

% Resolução

optimize(LMIs,[],sdpsettings('verbose',0,'solver','sedumi'));

res = min(checkset(LMIs));

% Número de Variáveis e Linhas

V = size(getvariables(LMIs),2);

linhas = vertices\*linhas; % LMIs totais

if res > 0

feas = 1;

K = double(Z)\*inv(double(X)); %Acha um K(a) com vértices

else

feas = 0;

K = NaN;

end

end