UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTE DE SISTEMAS E ENERGIA

IA–892 LISTA 2018 – PARTE 1

Giovanni Chemello Caprio
RA: 211483

1.

O primeiro exemplo foi gerado na forma contínua e com quatro estados do sistema e quatro vértices.

Na geração do grid, o maior valor real foi obtido em 2.4448. Esse valor foi encontrado no grid entre os vértices A_2 e A_4, com $a^* = (0.54\ 0.46)$.

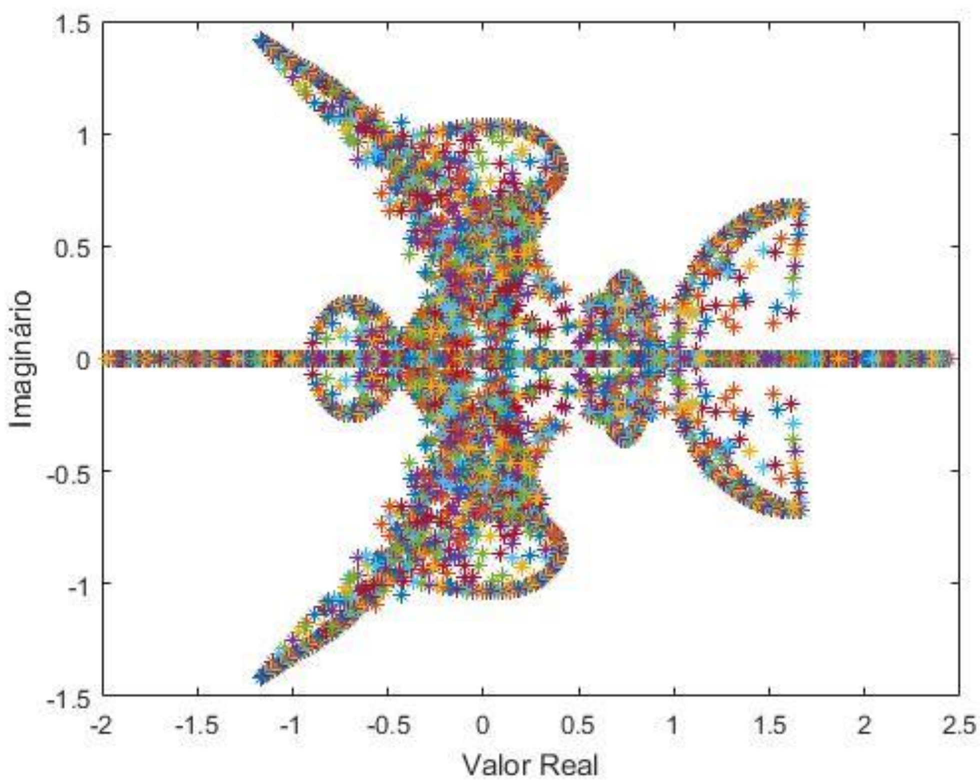
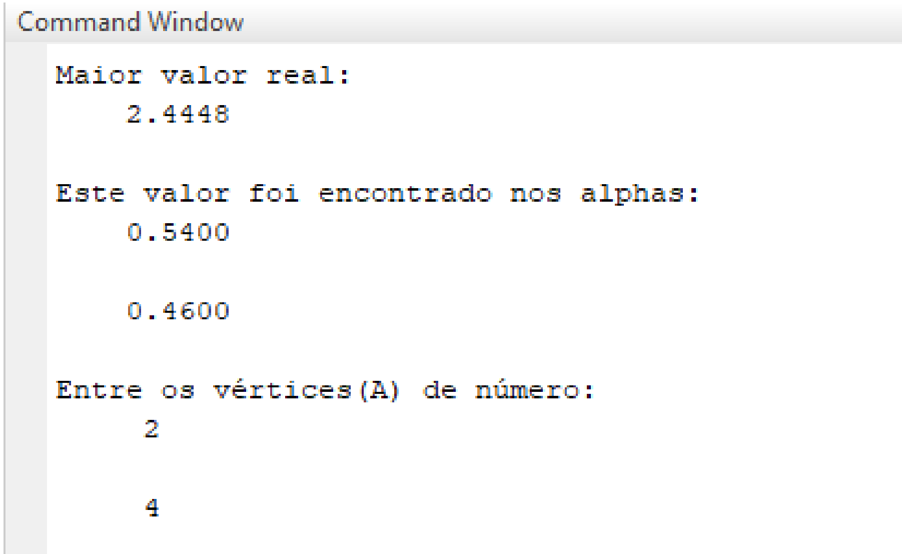


Figure 1 Lugar das raízes Caso Continuo



```
Command Window
Maior valor real:
    2.4448

Este valor foi encontrado nos alphas:
    0.5400

    0.4600

Entre os vértices(A) de número:
     2

     4
```

Figure 2. Resposta do MATLAB ao Caso Contínuo

O segundo exemplo foi gerado na forma discreta e com três estados do sistema e quatro vértices.

Na geração do grid, o maior valor real foi obtido em 2.7402. Esse valor foi encontrado no grid entre os vértices A_1 e A_3, com $a^* = (0.23\ 0.77)$.

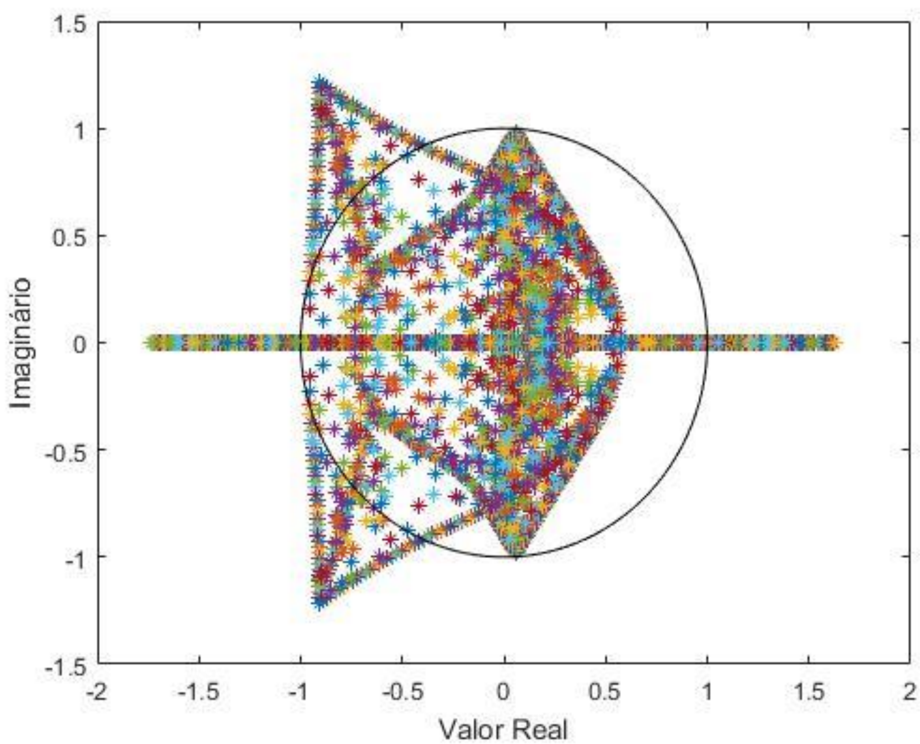


Figure 3 Lugar das raízes Caso Discreto

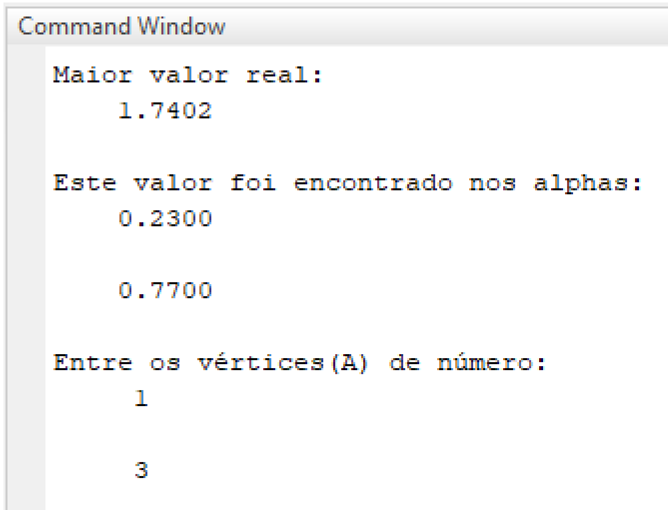


Figure 4. Lugar das raízes no Caso Discreto

- Código:

```
    %% Exercicio 1

    clc
    clear all
    close all

    %% Entrada

n = 3;       % numero de estados do sistema
N = 4;       % numero de vertices
caso =  1;   % 0 = continuo ; 1 = discreto;
for i = 1:N                             % criação dos A(a) para teste
    A{i} = randn(n);
end

    %% 1000 pontos uniformemente distribuidos

q = 0; %verificação final
maxEig = -10^10;
for i = 1:1000
    Aa = zeros(n);
    %criação dos pontos uniformes
    x(1)=1-rand^(1/(N-1));
        for k=2:N-1
            x(k)=(1-sum(x(1:k-1)))*(1-rand^(1/(N-k)));
        end
    x(N) = 1-sum(x(1:N-1));
        for t = 1:N
            Aa = Aa + (x(t)*A{t});
        end
    autoValores = eig(Aa);
        %maior valor real
```

```
        if caso == 0
            maximo = max(real(autoValores));
        else
            maximo = max(abs(autoValores));
        end
        if maximo > maxEig
            maxEig = maximo;
            q = 0;
            for s = 1:N
                g(s) = x(s);
            end
        end
    plot(real(autoValores),imag(autoValores),'*')
    xlabel('Valor Real')
    ylabel('Imaginário')
    hold on
end


    %% 100 pts igualmente espaçados em 3 vertices

for t = 1:N %varre os A
    for k = (t+1):N
        if t~=k %verifica se não é o mesmo vértice
            for i = 0:0.01:1
                Aa=i*A{t}+(1-i)*A{k};
                autoValores = eig(Aa);
                plot(real(autoValores),imag(autoValores),'*')
                hold on
                %maior valor real
                if caso == 0
                    maximo = max(real(autoValores));
                else
                    maximo = max(abs(autoValores));
                end
                if maximo > maxEig
                    maxEig = maximo;
                    q = 1;
                    g(1) = t;
                    g(2) = k;
                    g(3) = i;
                    g(4) = 1-i;
                end
            end
        end
    end
end

    %% 150 pts uniformemente distribuidos em 3 vertices

if N>3  %verifica se os vértices são maiores que 3
        for t = 1:N     %Varre A
            for k = (1+t):N
                    for l = (1+k):N
                        if (l~=k||l~=t||t~=k)     %só verifica se forem 3
vértices diferentes
```

```
                              for p = 1:150
                                   %criação dos pontos uniformes
                                   test = 2;
                                   while test > 1
                                       z(1) = (rand);
                                       z(2) = rand;
                                       test = sum(z(1:2));
                                   end
                                   z(3) = 1-sum(z(1:2));
                                   Aa= z(1)*A{t}+z(2)*A{k}+z(3)*A{l};
                                   autoValores = eig(Aa);
                                   %maior valor real
                                   if caso == 0
                                       maximo = max(real(autoValores));
                                   else
                                       maximo = max(abs(autoValores));
                                   end
                                   if maximo > maxEig
                                       maxEig = maximo;
                                       q = 2;
                                       g(1) = t;
                                       g(2) = k;
                                       g(3) = l;
                                       g(4) = z(1);
                                       g(5) = z(2);
                                       g(6) = z(3);
                                   end
                                   plot(real(autoValores),imag(autoValores),'*')
                                   hold on
                              end
                           end
                       end
               end
           end
else
    disp('Número de vértices menor que 4')
end

%% Lógica para display dos resultados

disp('Maior valor real:' )
disp(maxEig)

disp('Este valor foi encontrado nos alphas:' )

if q == 0
   for s = 1:N
       disp(g(s));
   end
end
if q == 1
    disp(g(3))
    disp(g(4))
    disp('Entre os vértices(A) de número:' )
    disp(g(1))
    disp(g(2))
```

```
end
if q == 2
    disp(g(4))
    disp(g(5))
    disp(g(6))
    disp('Entre os vértices(A) de número:' )
    disp(g(1))
    disp(g(2))
    disp(g(3))
end
if caso == 1
    tetha=-pi:.055:pi;
    xr=cos(tetha);
    yr=sin(tetha);
    plot(xr,yr,'black');
end
```

2. Após algumas buscas no código, um exemplo foi gerado com maior módulo tendo o valor de 0.9248.

   Esse exemplo apresentou 1.8513 para a norma H_2 e 4.5699 para a norma H_{inf}.

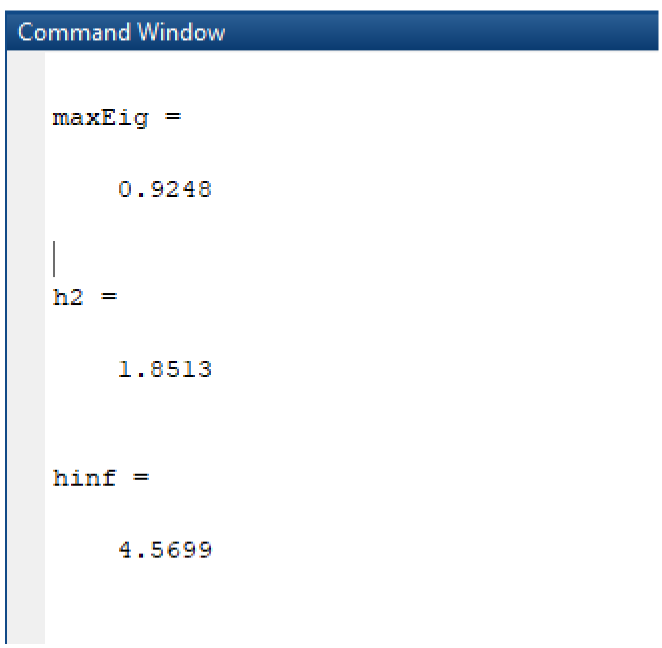


Figure 5 Resultado do Matlab

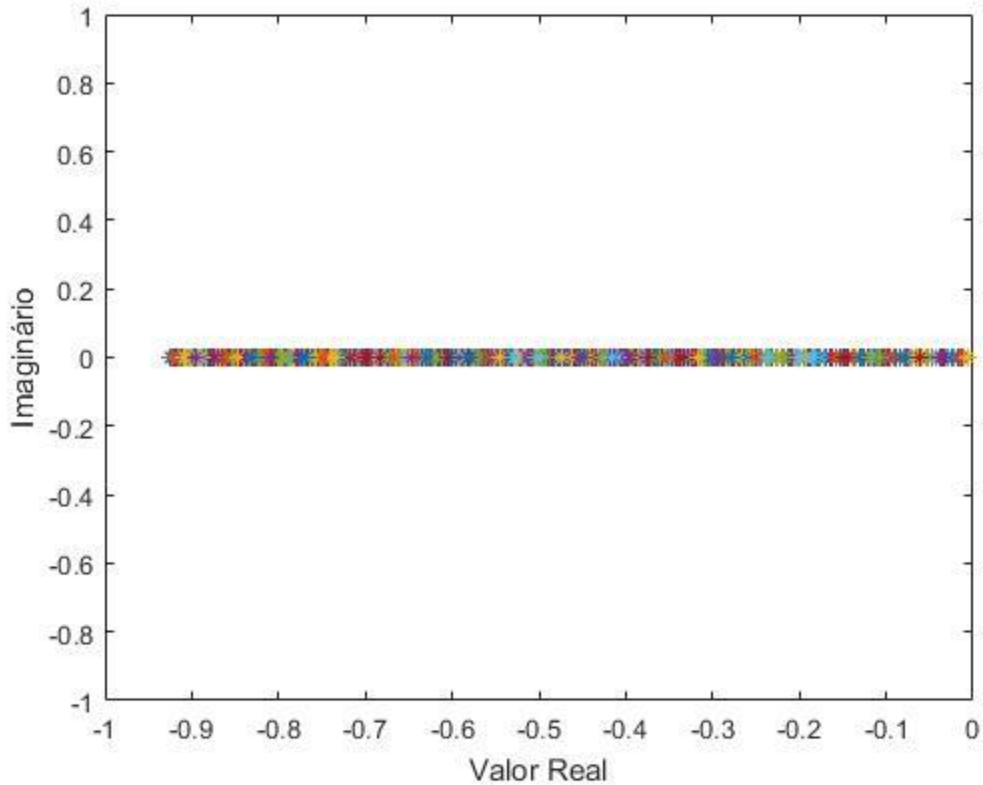


Figure 6 Lugas das raízes

- Código:

```
%% Exercicio 2
clc, clear all, close all

% Entradas

n = 3;          % numero de estados
m = 2;          % numero de entradas
l = 1;          % numero de saidas
N = 4;          % numero de vértices


% Cria os A estáveis (Autovalores negativos com d < 1, diagonal princiapal)
for i = 1:N                 % A = n x n
    A{i} = zeros(n);
    for x = 1:n
        A{i}(x,x) = -rand;
    end
end

% Cria B
for i = 1:N                 % B = n x m
    B{i}(1,1) = rand;
```

```
    for x = 1:n
        for t = 1:m
            B{i}(x,t) = rand;
        end
    end
end

% Cria C
for i = 1:N                % B = l x n
    C{i}(1,1) = rand;
    for x = 1:l
        for t = 1:n
            C{i}(x,t) = rand;
        end
    end
end

% Cria D
for i = 1:N                % D = l x m
    D{i}(1,1) = rand;
    for x = 1:l
        for t = 1:m
            D{i}(x,t) = rand;
        end
    end
end

q = 0; %verificação final
h2 = 0;
hinf = 0;
maxEig = -10^10;

%%   GRID


    % 1000 pontos uniformemente distribuidos

q = 0; %verificação final
maxEig = -10^10;
for i = 1:1000
    A_ = zeros(n);
    B_ = zeros(n,m);
    C_ = zeros(l,n);
    D_ = zeros(l,m);
    %criação dos pontos uniformes
    x(1)=1-rand^(1/(N-1));
        for k=2:N-1
            x(k)=(1-sum(x(1:k-1)))*(1-rand^(1/(N-k)));
        end
    x(N) = 1-sum(x(1:N-1));
        for t = 1:N
            A_ = A_ + (x(t)*A{t});
            B_ = B_ + (x(t)*B{t});
            C_ = C_ + (x(t)*C{t});
            D_ = D_ + (x(t)*D{t});
```

```
        end
        sys = ss(A_,B_,C_,D_,-1);
        h2_ = norm(sys,2);
        hinf_ = norm(sys,inf);
        autoValores = eig(A_);
        plot(real(autoValores),imag(autoValores),'*')
        xlabel('Valor Real')
        ylabel('Imaginário')
        hold on
        maximo = max(abs(autoValores));
        if maximo > maxEig
            maxEig = maximo;
            q = 0;
        end
        if h2_ > h2
            h2 = h2_;
        end
        if hinf_ > hinf
            hinf = hinf_;
        end
end


    %% 100 pts igualmente espaçados em 3 vertices

autoValores = 0;
for t = 1:N %varre os A
    for k = (t+1):N
        if t~=k %verifica se não é o mesmo vértice
            for i = 0:0.01:1
                A_=i*A{t}+(1-i)*A{k};
                B_=i*B{t}+(1-i)*B{k};
                C_=i*C{t}+(1-i)*C{k};
                D_=i*D{t}+(1-i)*D{k};
                autoValores = eig(A_);
                plot(real(autoValores),imag(autoValores),'*')
                hold on
                sys = ss(A_,B_,C_,D_,-1);
                h2_ = norm(sys,2);
                hinf_ = norm(sys,inf);
                %maior valor real
                maximo = max(abs(autoValores));
                if maximo > maxEig
                    maxEig = maximo;
                    q = 0;
                end
                if h2_ > h2
                    h2 = h2_;
                end
                if hinf_ > hinf
                    hinf = hinf_;
                end
            end
        end
    end
end
```

```matlab
    %% 150 pts uniformemente distribuidos em 3 vertices
if N>3  %verifica se os vértices são maiores que 3
        for t = 1:N     %Varre A
            for k = (1+t):N
                    for l = (1+k):N
                        if (l~=k||l~=t||t~=k)     %só verifica se forem 3 vértices
diferentes
                           for p = 1:150
                                %criação dos pontos uniformes
                                test = 2;
                                while test > 1
                                    z(1) = (rand);
                                    z(2) = rand;
                                    test = sum(z(1:2));
                                end
                                z(3) = 1-sum(z(1:2));
                                A_ = z(1)*A{t}+z(2)*A{k}+z(3)*A{l};
                                B_ = z(1)*B{t}+z(2)*B{k}+z(3)*B{l};
                                C_ = z(1)*C{t}+z(2)*C{k}+z(3)*C{l};
                                D_ = z(1)*D{t}+z(2)*D{k}+z(3)*D{l};
                                autoValores = eig(A_);
                                %maior valor real
                                maximo = max(abs(autoValores));
                                if maximo > maxEig
                                    maxEig = maximo;
                                    q = 0;
                                end
                                if h2_ > h2
                                    h2 = h2_;
                                end
                                if hinf_ > hinf
                                    hinf = hinf_;
                                end
                                plot(real(autoValores),imag(autoValores),'*')
                                hold on
                           end
                        end
                    end
            end
        end
else
    disp('Número de vértices menor que 4')
end

%% Saidas

maxEig
h2
hinf
for i = 1:N
   A{i}
   B{1}
   C{1}
   D{1}
end
```

3. Anexado no Final (Escrito à mão)

4. Anexado no Final (Escrito à mão)

5.

a)

Nesta questão foi analisado os sinais de sáida e entrada e ao computar sua relação, o seu resultado foi de 4.3925.

b)

Computando a norma H infitito por LMI, obtivemos:

```
Command Window

Linear matrix variable 1x2 (full, real, 3 variables)

ans =

  struct with fields:

    cpusec_m: 1.3100
           V: 4
    cpusec_s: 1.3358
       delta: -3.4955e-11
        feas: 1
           P: [2×2 double]
        hinf: 10.1934
```

E variando o w para o programa, obtivemos os seguintes valores:

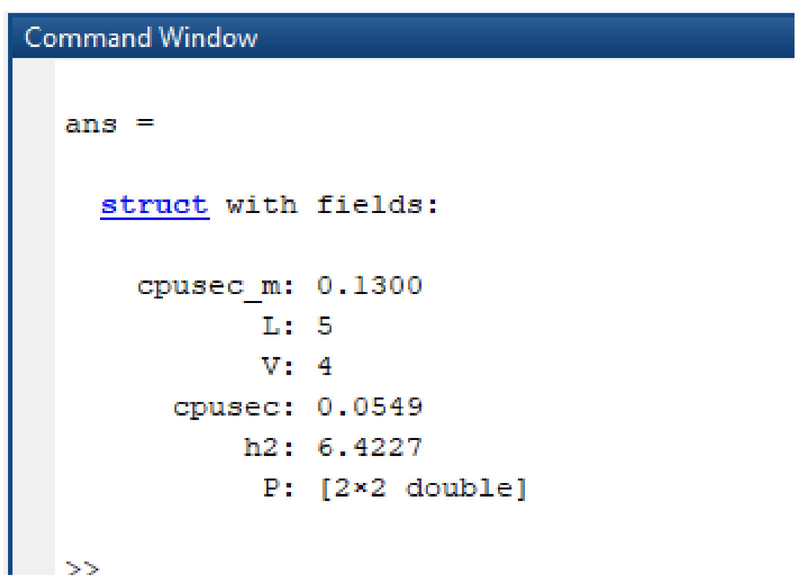
w	relação
3.80	8.0550
3.85	8.1988
3.90	8.2637
3.95	8.2598
4.00	8.2141
4.05	8.1488
4.10	8.0657
4.15	7.9491
4.20	7.7818

c)

Utilizando a entrada como impulso, chegou em 2.3747 para o valor da norma H_2.

d)

Computando a norma H_2 por LMI, obtivemos:



```
Command Window

ans =

  struct with fields:

    cpusec_m: 0.1300
           L: 5
           V: 4
      cpusec: 0.0549
          h2: 6.4227
           P: [2×2 double]

>>
```

Figure 7 Resultado da norma H2 por LMI

- Código:

```
%% Exercicio 5
clc,clear all,close all

% Dados

A = [0 1; -16 -0.8];
B = [1;1];
C = [1 2];
D = 0;

sys = ss(A,B,C,D);
%% A)

% Cálculo da energia de Entrada

w = 3;
T = 0.01;
t = 0:T:10;
sinal_entrada = sin(w*t).*exp(-0.1*t);
energia_entrada = sqrt(trapz(sinal_entrada.^2)*T);

% Cálculo da energia de Saída

sinal_saida = lsim(sys,sinal_entrada,t);  % sinal de saida dependente de u,t,sys
energia_saida = sqrt(trapz(sinal_saida.^2)*T);

% Cáculo da relação entre Saída/Entrada
relacao =  energia_saida/energia_entrada


%% B)

infinito(A,B,C,D) % Roda a função que cálcula a LMI e informa a norma Hinf

% Verificação da simulação com a nomr Hinf
for k = 0:0.05:0.4

    w = 3.8+k;
    T = 0.01;
    t = 0:T:10;
    sinal_entrada = sin(w*t).*exp(-0.1*t);
    energia_entrada = sqrt(trapz(sinal_entrada.^2)*T);

    % Cálculo da energia de Saída
```

```
    sinal_saida = lsim(sys,sinal_entrada,t);  % sinal de saida dependente de
u,t,sys
    energia_saida = sqrt(trapz(sinal_saida.^2)*T);

    % Cáculo da relação entre Saída/Entrada
    disp('A relação entre a energia de saída e entrada é:')
    relacao_total =  energia_saida/energia_entrada
    w

end

%% C)

% Cálculo da energia de Saída

sinal_saida = impulse(sys);  % sinal de saida dependente de u,t,sys
energia_H2 = sqrt(trapz(sinal_saida).^2*T)



%% D)

H2(A,B,C,D) % Roda a função que cálcula a LMI e informa a norma H2
```

- Função H_{inf} :

```
function output = hinf_norm_c_yal(A,B,C,D)

% Configurações
hinf = 0;
tol = 1e-7;
order = size(A,1);
inputs = size(B,2);
outputs = size(C,1);
output.cpusec_m = clock;

% Criação das LMIs
LMIs = [];
mu = sdpvar(1);
obj = mu;

P = sdpvar(order,order,'symmetric'); % Lyapunov > 0
LMIs = [LMIs, P >= 0];

% Bounded real lemma  < 0
T11 = A'*P + P*A;
T12 = P*B;
T13 = C';
T22 = -eye(inputs);
T21 = B'*P
T23 = D';
```

```
T33 = -mu*eye(outputs);
T = [T11 T12 T13;
    T12' T22 T23;
    T13' T23' T33];
LMIs = [LMIs, T <= 0];

output.cpusec_m = etime(clock,output.cpusec_m);

% Número de Variáveis
output.V = size(getvariables(LMIs),2);

% Solução a LMIs
sol = solvesdp(LMIs,obj,sdpsettings('verbose',0,'solver','sedumi'));

% Tempo para solução
output.cpusec_s = sol.solvertime;

% Resíduo
p = min(checkset(LMIs));
output.delta = p;
output.feas = 0;

% Soluções, se existirem
if p > -tol
    output.P = double(P);
    output.hinf = sqrt(double(mu));
    output.feas = 1;
end
```

- Função H_2 :

```
function out = h2_lmi_c(A,B,C,param)

% Configurações
h2 = 0;
precision = 1e-7;
order = size(A,1);
inputs = size(B,2);
outputs = size(C,1);
out.cpusec_m = clock;

% Criação das LMIs
LMIs = [];
out.L = 0;    % LMI contagem de linhas
mu = sdpvar(1);
obj = mu;

P = sdpvar(order,order,'symmetric');    % Lyapunov > 0
LMIs = [LMIs, P >= 0];
out.L = out.L + order;
```

```matlab
% Condição do Traço (mu > trace( B' P B) )
LMIs = [LMIs, mu > trace(B'*P*B)];

% Graminiano(A P + P A' + C'C < 0)
LMIs = [LMIs, A'*P + P*A + C'*C <= 0];
out.L = out.L + order + 1;
out.cpusec_m = etime(clock,out.cpusec_m);

% Número de Variáveis
out.V = size(getvariables(LMIs),2);

% Solução a LMIs
sol = solvesdp(LMIs,obj,sdpsettings('verbose',0,'solver','sedumi'));

% Tempo para solução
out.cpusec = sol.solvertime;

% Resíduo
p=min(checkset(LMIs));
out.h2 = 0;

% Soluções, se existirem
if p > -precision
    out.h2 = sqrt(double(mu));
    out.P = double(P);
end
```