

# COE-835 Controle adaptativo

## Trabalho 10

**Grupo:** Guilherme Pires Sales de Carvalho  
Matheus Ferreira dos Reis  
Renan Salles de Freitas

**Algoritmo:** Adaptive Backstepping Control

**Caso:**  $n = 3$  (ordem da planta)  
 $n^* = 3$  (grau relativo)  
 $n_p = 4$  (# de parâmetros)

## Conteúdo

<b>1</b>	<b>Backstepping - Formulação teórica com observador de ordem completa</b>	<b>2</b>
<b>2</b>	<b>Implementação</b>	<b>4</b>
<b>3</b>	<b>Resultados das simulações</b>	<b>8</b>
3.1	Simulação #1 . . . . .	8
3.2	Simulação #2 . . . . .	11
3.3	Simulação #3 . . . . .	12
<b>4</b>	<b>Discussão</b>	<b>16</b>

## 1 Backstepping - Formulação teórica com observador de ordem completa

Neste trabalho, consideramos o sistema:

$$\begin{aligned}\dot{x}_1 &= -a_2 x_1 + x_2 \\ \dot{x}_2 &= -a_1 x_1 + x_3 \\ \dot{x}_3 &= -a_0 x_1 + k_p u \\ y &= x_1\end{aligned}\tag{1}$$

onde os parâmetros  $a_2$ ,  $a_1$ ,  $a_0$  e  $k_p$  são desconhecidos. Para esta formulação apenas a saída do sistema  $y$  está disponível, portanto  $x_2$  e  $x_3$  não são conhecidos e devem ser estimados. Podemos reescrever o sistema 2:

$$\begin{aligned}\dot{x} &= Ax + F(y, u)^\top \theta \\ A &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, F(y, u)^\top = [B(u) \quad \Phi(y)], \Phi(y) = \begin{bmatrix} -y & 0 & 0 \\ 0 & -y & 0 \\ 0 & 0 & -y \end{bmatrix}, B(u) = \begin{bmatrix} 0 \\ 0 \\ u \end{bmatrix}, \theta = \begin{bmatrix} k_p \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \\ y &= e_1^\top x \\ e_1 &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\end{aligned}\tag{2}$$

Para estimar os estados, utilizamos os filtros abaixo:

$$\begin{aligned}\dot{\xi} &= A_0 \xi + ky \\ \dot{\Omega}^\top &= A_0 \Omega^\top + F^\top \\ k &= \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix}, A_0 = A - k e_1^\top = \begin{bmatrix} -k_1 & 1 & 0 \\ -k_2 & 0 & 1 \\ -k_3 & 0 & 0 \end{bmatrix}\end{aligned}\tag{3}$$

Os valores de  $k$  devem ser escolhidos de forma que  $A_0$  seja Hurwitz. E, dessa forma, o estado estimado pode ser escrito como:

$$\hat{x} = \xi + \Omega^\top \theta\tag{4}$$

Derivando a equação 4 e substituindo as equações dos filtros 3, verifica-se que a dinâmica do estimador é igual à dinâmica da planta 2.

Porém,  $\Omega$  é uma matriz e opta-se pela redução das ordens dos filtros. Observe que  $\Omega^\top = [v_0 \quad | \quad \Xi]$  e, pela equação 3, temos que:

$$\dot{v}_0 = A_0 v_0 + e_3 u\tag{5}$$

$$\dot{\Xi} = A_0 \Xi - I y\tag{6}$$

Introduzem-se dois novos filtros, para substituir os filtros da equação 3:

$$\dot{\lambda} = A_0\lambda + e_3u \quad (7)$$

$$\dot{\eta} = A_0\eta + e_3y \quad (8)$$

É fácil verificar que, para esta planta de segunda ordem sem zeros ( $m = 0$ ),  $v_0 = \lambda$ . Para o caso geral, temos que:

$$\dot{\lambda} = A_0\lambda + e_3u \quad (9)$$

$$v_i = A_0^i\lambda \quad (i = 0, \dots, m) \quad (10)$$

É possível demonstrar que:

$$\Xi = -[A_0^2\eta \quad A_0\eta \quad \eta] \quad (11)$$

$$\xi = -A_0^3\eta \quad (12)$$

Podemos reescrever a dinâmica da saída  $y$ :

$$\dot{y} = k_p v_{0,2} + \xi_2 + \bar{\omega}^\top \theta + \epsilon_2 \quad (13)$$

$$\bar{\omega}^\top = [0 \quad (\Xi_2 + \phi_1^\top)]$$

Desta forma, o sistema 2 pode ser representado com os estados do observador:

$$\dot{y} = k_p v_{0,2} + \xi_2 + \bar{\omega}^\top \theta + \epsilon_2 \quad (14)$$

$$\begin{aligned} \dot{v}_{0,2} &= v_{0,3} - k_2 v_{0,1} \\ \dot{v}_{0,3} &= -k_3 v_{0,1} + u \end{aligned} \quad (15)$$

O projeto backstepping agora segue como na seção anterior. Primeiro, fazemos a mudança de coordenadas em  $\mathbf{z}$ :

$$z_1 = y - y_r \quad (16)$$

$$z_2 = v_{0,2} - \alpha_1 - \hat{\rho} \dot{y}_r$$

$$z_3 = v_{0,3} - \alpha_2 - \hat{\rho} \ddot{y}_r$$

onde  $\rho$  é estimativa de  $\frac{1}{k_p}$ . Os controles virtuais  $\alpha_1$  e  $\alpha_2$ , a lei de controle  $u$  e as leis de adaptação  $\dot{\theta}$  e  $\dot{\rho}$  são obtidas pelo método de Lyapunov. A formulação e desenvolvimento dos passos da solução do problema são realizados nas notas de aula. Dessa forma, neste relatório fazemos o resumo das equações:

$$\begin{aligned}
\alpha_1 &= \hat{\rho}\bar{\alpha}_1 = \hat{\rho}(-c_1z_1 - d_1z_1 - \xi_2 - \bar{\omega}^\top\hat{\theta}) \\
\dot{\hat{\rho}} &= -\gamma \text{sign}(k_p) [\dot{y}_r + \bar{\alpha}_1] z_1 \\
\tau_1 &= [\omega - \hat{\rho}(\dot{y}_r + \bar{\alpha}_1)e_1] z_1 \\
\beta_2 &= k_2 v_{0,1} + \frac{\partial \alpha_1}{\partial y}(\xi_2 + \omega^\top\hat{\theta}) + \frac{\partial \alpha_1}{\partial \eta}(A_0\eta + e_3y) + \frac{\partial \alpha_1}{\partial y_r}\dot{y}_r + \left(\dot{y}_r + \frac{\partial \alpha_1}{\partial \hat{\rho}}\right)\dot{\hat{\rho}} \\
\tau_2 &= \tau_1 - \frac{\partial \alpha_1}{\partial y}\omega z_2 \\
\alpha_2 &= -c_2z_2 - \hat{k}_p z_1 + \beta_2 + \frac{\partial \alpha_1}{\partial \hat{\theta}}\Gamma\tau_2 - d_2 \left(\frac{\partial \alpha_1}{\partial y}\right)^2 z_2 \\
\beta_3 &= k_3 v_{0,1} + \frac{\partial \alpha_2}{\partial y}(\xi_2 + \omega^\top\hat{\theta}) + \frac{\partial \alpha_2}{\partial \eta}(A_0\eta + e_3y) + \frac{\partial \alpha_2}{\partial \lambda_1}(-k_1\lambda_1 + \lambda_2) + \frac{\partial \alpha_2}{\partial \lambda_2}(-k_2\lambda_2 + \lambda_3) + \\
&\quad + \frac{\partial \alpha_2}{\partial y_r}\dot{y}_r + \frac{\partial \alpha_2}{\partial \hat{y}_r}\ddot{y}_r + \left(\ddot{y}_r + \frac{\partial \alpha_2}{\partial \hat{\rho}}\right)\dot{\hat{\rho}} \\
\tau_3 &= \tau_2 - \frac{\partial \alpha_2}{\partial y}\omega z_3 \\
\dot{\hat{\theta}} &= \Gamma\tau_3 \\
u &= -c_3z_3 + \beta_3 + \hat{\rho}\ddot{y}_r + \frac{\partial \alpha_2}{\partial \hat{\theta}}\Gamma\tau_3 - d_3 \left(\frac{\partial \alpha_2}{\partial y}\right)^2 z_3 - z_2 \frac{\partial \alpha_1}{\partial \hat{\theta}}\Gamma \frac{\partial \alpha_2}{\partial y}\omega
\end{aligned}$$

## 2 Implementação

Para a implementação, só é preciso computar os filtros:

$$\dot{\lambda} = A_0\lambda + e_3u \quad (17)$$

$$\dot{\eta} = A_0\eta + e_3y \quad (18)$$

Mas é preciso, ainda, computar todas as derivadas parciais das variáveis de controle virtuais. Essas derivadas podem ser calculadas previamente ou pode-se utilizar MatLab simbólico e pedir ao programa a solução das equações parciais. O MatLab, porém, ainda não é capaz de realizar eficientemente derivadas parciais em vetores e não simplifica as equações de forma a otimizar o desempenho. Verificou-se ainda, que muitas vezes ocorre overflow indevido, justamente por causa da não simplificação das equações. Dessa forma, optou-se por derivar cada equação. Todas as derivadas parciais estão representadas nas linhas de código abaixo:

```

1 %-----%
2 %
3 % COE-835 Controle adaptativo
4 %
5 % Script para simular o trabalho 10
6 %
7 % Backstepping : n = 3 Second and third order plant
8 %                 n* = 3 Relative degree
9 %                 np = 4 Adaptive parameters
10 % Caso com observador completo
11 %-----%
12
13 function dx = backstepping_obs(t,x)
14
15 global A thetas A0 c1 c2 c3 d1 d2 d3 Gamma gamma kp a w e1 e2 e3 k;

```

```

16
17 X           = x(1:3); y = e1'*X;
18 theta       = x(4:7);
19 lambda     = x(8:10);
20 eta         = x(11:13);
21 rho         = x(14);
22
23 %% Input
24 yr=0; dyr=0; ddyr=0; dddyr=0;
25 for i=1:length(a)
26     yr = yr + a(i)*sin(w(i)*t);
27     dyr = dyr + w(i)*a(i)*cos(w(i)*t);
28     ddyr = ddyr - w(i)^2*a(i)*sin(w(i)*t);
29     dddyr = dddyr - w(i)^3*a(i)*cos(w(i)*t);
30 end
31
32 Phi = [-y 0 0; 0 -y 0; 0 0 -y];
33
34 %% Variables 1
35 xi = -A0^3 * eta;
36 Xi = -[A0^2*eta A0*eta eta];
37 v0_1 = lambda(1);
38 v0_2 = lambda(2);
39 v0_3 = lambda(3);
40 omega_bar = [0, (Xi(2,:)-y*e1')];
41 omega = [v0_2, (Xi(2,:)-y*e1')];
42
43 %% Z
44 z1 = y - yr;
45 alpha_bar = -c1*z1 - d1*z1 - xi(2) - omega_bar'*theta;
46 alpha_1 = rho * alpha_bar;
47 z2 = v0_2 - rho*dyr - alpha_1;
48
49 %% Filtro eta
50 deta = A0*eta + e3*y;
51
52 %% dalpha/dt
53 dady = rho * (- c1 - d1 + [0,e1']*theta);
54 dadeta_deta = rho * (e2' * A0^3 * deta + ...
55     [0,e2'*A0^2*deta, e2'*A0*deta, e2'*eye(3)*deta]*theta);
56 dadyr = rho*(c1 + d1);
57 dadtheta = - rho * omega_bar';
58 dadrho = -(c1 + d1)*z1 - e2'*xi - omega_bar'*theta;
59
60 %% dz2/dt
61 dz2dy = - dady;
62 dz2deta_deta = -dadeta_deta;
63 dz2dyr = - dadyr;
64 dz2dtheta = - dadtheta;
65 dz2drho = - dyr - dadrho;
66 dz2dlambda1 = 0;
67 dz2dlambda2 = 1;
68 dz2ddyr = - rho;
69
70 %% dz1/dt
71 dz1dy = 1;
72 dz1deta_deta = 0;
73 dz1dyr = -1;
74 dz1dtheta = 0;

```

```

75 dz1drho = 0;
76 dz1dlambda1 = 0;
77 dz1dlambda2 = 0;
78 dz1ddyr = 0;
79
80 %% ddrho/dt
81 ddrhody = - gamma*sign(kp)*((dyr+alpha_bar) + z1*dady/rho);
82 ddrhodata = - gamma*sign(kp)*z1*dadeta_deta/rho;
83 ddrhodtyr = - gamma*sign(kp)*(-1*(dyr+alpha_bar) + dadtyr/rho);
84 ddrhodtheta = - gamma*sign(kp)*z1*dadtheta/rho;
85 ddrhodrho = 0;
86 ddrhodlambda1 = 0;
87 ddrhodlambda2 = 0;
88 ddrhoddyr = - gamma*sign(kp)*z1;
89
90 %% Variables 2
91 tau_1 = (omega - rho*(dyr + alpha_bar)*[e1',0]')*z1;
92 tau2 = tau_1 - z2 * (dady * omega);
93
94 %% Atualização 1
95 drho = - gamma * z1 * sign(kp) * (dyr + alpha_bar);
96 beta_2 = k(2)*v0_1 + dady * (xi(2) + omega'*theta) + ...
97     dadeta_deta + dadtyr * dyr + (dyr + dadrho) * drho;
98 alpha2 = -c2*z2 + beta_2 + dadtheta*Gamma*tau2 - d2*z2*(dady)^2 - ...
99     z1*theta(1);
100
101 %% dbeta2/dt
102 dadeta_dy = rho * (e2' * A0^3 * e3 + ...
103     [0, e2'*A0^2*e3, e2'*eye(3)*e3]*theta);
104 dbeta2dy = dady*([0, e1']*theta) + dadeta_dy + ...
105     drho*(-(c1+d1)-[0, e1']*theta) + (dyr+dadrho)*ddrhody;
106 dbeta2deta_deta = dady*dadeta_deta/rho + ...
107     [0, e2'*A0^3*data, e2'*A0^2*data, e2'*A0*data]*theta + ...
108     drho*(-dadeta_deta/rho) + (dyr + dadrho)*ddrhodata;
109 dbeta2dtyr = drho*(c1+d1) + (dyr + dadrho)*ddrhodtyr;
110 dbeta2dlambda1 = k(2);
111 dbeta2dlambda2 = dady*[1, 0, 0, 0]*theta;
112 dbeta2drho = (dady/rho)*(xi(2)+omega'*theta) + dadeta_deta/rho + ...
113     (dadtyr/rho)*dyr;
114 dbeta2ddyr = dadtyr + drho + (dyr+dadrho)*ddrhoddyr;
115 dbeta2dtheta = rho*[0, e1']*(xi(2)+omega'*theta) + dady*omega' + ...
116     rho*[0, e2'*A0^2*data, e2'*A0*data, e2'*eye(3)*data] - ...
117     drho*omega_bar' + (dyr+dadrho)*ddrhodtheta;
118
119 %% dtau2/dt
120 dtau2dy = [0, e1']'*z1 + omega - dady*[e1', 0]'*z1 - ...
121     rho*(dyr+alpha_bar)*[e1', 0]' - z2*dady*[0, e1']' + (dady)^2*omega;
122 dtau2deta_deta = [0, e2'*A0^2*data, e2'*A0*data, e2'*eye(3)*data]'*z1 - ...
123     dadeta_deta*[e1', 0]'*z1 + dadeta_deta*dady*omega - ...
124     z2*dady*[0, e2'*A0^2*data, e2'*A0*data, e2'*eye(3)*data]';
125 dtau2dtyr = -omega - dadtyr*[e1', 0]'*z1 + rho*(dyr+alpha_bar)*[e1', 0]' + ...
126     dadtyr*dady*omega;
127 dtau2dlambda1 = [1, 0, 0, 0]'*z1 - z2*dady*[1, 0, 0, 0]';
128 dtau2dlambda2 = -dady*omega;
129 dtau2drho = -(dyr+alpha_bar)*[e1', 0]'*z1 - (-dyr-dadrho)*dady*omega - ...
130     z2*(dady)/rho*omega;
131 dtau2ddyr = -rho*[e1', 0]'*z1 + rho*dady*omega;
132 dtau2dtheta = 0;
133

```

```

134 %% dadtheta/dt
135 dadthetady = [0,0,0,0];
136 dadthetadeta_deta = -rho*[0,e2'*A0^2*deta, e2'*A0*deta, e2'*eye(3)*deta];
137 dadthetadtyr = [0,0,0,0];
138 dadthetadlambda1 = [0,0,0,0];
139 dadthetadlambda2 = [0,0,0,0];
140 dadthetadrho = [-0,e2'*A0^2*eta, e2'*A0*eta, e2'*eye(3)*eta];
141 dadthetaddyr = [0,0,0,0];
142 dadthetadtheta = [0,0,0,0];
143
144 %% (dady)^2/dt
145 dady2dy = 0;
146 dady2deta_deta = 0;
147 dady2dtyr = 0;
148 dady2dlambda1 = 0;
149 dady2dlambda2 = 0;
150 dady2drho = 2*rho*(- c1 - d1 + [0,e1']*theta)^2;
151 dady2ddyr = 0;
152 dady2dtheta = 2*(rho^2)*(- c1 - d1 + [0,e1']*theta)*[0,e1'];
153
154 %% da2/dt
155 da2dy = -c2*dz2dy - theta(1)*dz1dy + dbeta2dy + dadthetady*Gamma*tau2 + ...
156     dadtheta*Gamma*dtau2dy - d2*dady2dy*z2 - d2*dady^2*dz2dy;
157 da2deta_deta = -c2*dz2deta_deta - theta(1)*dz1deta_deta + ...
158     dbeta2deta_deta + dadthetadeta_deta*Gamma*tau2 + ...
159     dadtheta*Gamma*dtau2deta_deta - d2*dady2deta_deta*z2 - ...
160     d2*dady^2*dz2deta_deta;
161 da2dlambda1 = -c2*dz2dlambda1 - theta(1)*dz1dlambda1 + dbeta2dlambda1 + ...
162     dadthetadlambda1*Gamma*tau2 + dadtheta*Gamma*dtau2dlambda1 - ...
163     d2*dady2dlambda1*z2 - d2*dady^2*dz2dlambda1;
164 da2dlambda2 = -c2*dz2dlambda2 - theta(1)*dz1dlambda2 + dbeta2dlambda2 + ...
165     dadthetadlambda2*Gamma*tau2 + dadtheta*Gamma*dtau2dlambda2 - ...
166     d2*dady2dlambda2*z2 - d2*dady^2*dz2dlambda2;
167 da2dtyr = -c2*dz2dtyr - theta(1)*dz1dtyr + dbeta2dtyr + ...
168     dadthetadtyr*Gamma*tau2 + dadtheta*Gamma*dtau2dtyr - ...
169     d2*dady2dtyr*z2 - d2*dady^2*dz2dtyr;
170 da2ddyr = -c2*dz2ddyr - theta(1)*dz1ddyr + dbeta2ddyr + ...
171     dadthetaddyr*Gamma*tau2 + dadtheta*Gamma*dtau2ddyr - ...
172     d2*dady2ddyr*z2 - d2*dady^2*dz2ddyr;
173 da2drho = -c2*dz2drho - theta(1)*dz1drho + dbeta2drho + ...
174     dadthetadrho*Gamma*tau2 + dadtheta*Gamma*dtau2drho - ...
175     d2*dady2drho*z2 - d2*dady^2*dz2drho;
176 da2dtheta = -c2*dz2dtheta - theta(1)*dz1dtheta + dbeta2dtheta + ...
177     dadthetadtheta*Gamma*tau2 + dadtheta*Gamma*dtau2dtheta - ...
178     d2*dady2dtheta*z2 - d2*dady^2*dz2dtheta;
179
180 %% Variables 3
181 z3 = lambda(3) - rho*ddyr - alpha2;
182 tau3 = tau2 - da2dy*omega*z3;
183 dtheta = Gamma * tau3;
184 beta3 = k(3)*v0_1 + da2dy * (xi(2) + omega'*theta) + ...
185     da2deta_deta + da2dtyr * dtyr + (ddyr + da2drho) * drho + ...
186     da2dlambda1 * (-k(1)*lambda(1)+lambda(2)) + ...
187     da2dlambda2 * (-k(2)*lambda(2)+lambda(3)) + da2ddyr*ddyr;
188 u = -c3*z3 + beta3 + rho*ddyr + da2dtheta*Gamma*tau3 - ...
189     d3*(da2dy)^2*z3 - z2*dadtheta*Gamma*da2dy*omega;
190
191
192 %% Filtros

```

```

193 dlambda = A0*lambda + e3*u;
194
195 %% Planta
196 F = [e3*u Phi];
197 dX = A*X + F*thetas;
198
199 %% Translation
200 dx = [dX' dtheta' dlambd' deta' drho'];

```

### 3 Resultados das simulações

Nas simulações, procuramos avaliar o comportamento do sistema para as seguintes condições: (i) condições iniciais  $\theta(0)$  e  $y(0)$ ; (ii) Parâmetros da planta e do modelo; (iii) ganho de adaptação  $\Gamma$ .

Apresentaremos os resultados obtidos através de simulações no ambiente **Matlab/Simulink** e os discutiremos na próxima seção.

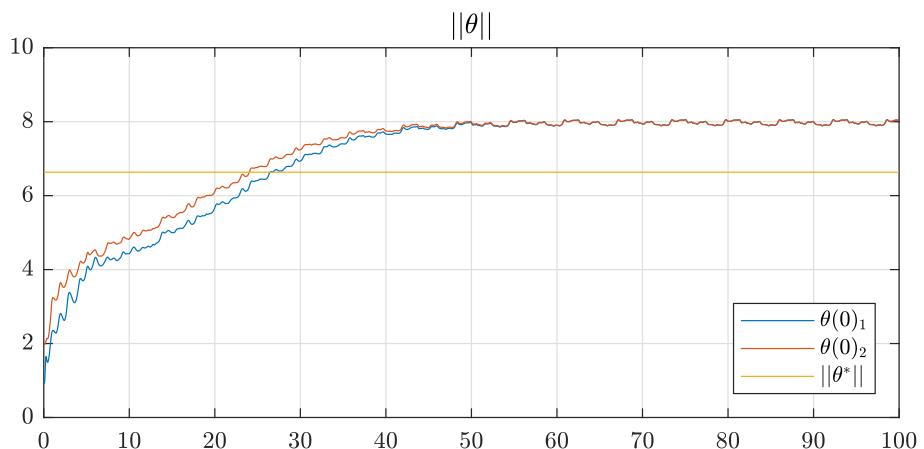
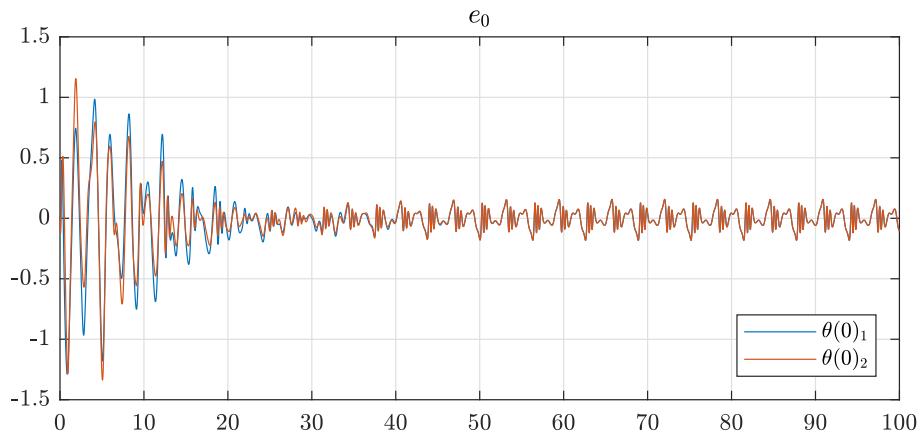
#### 3.1 Simulação #1

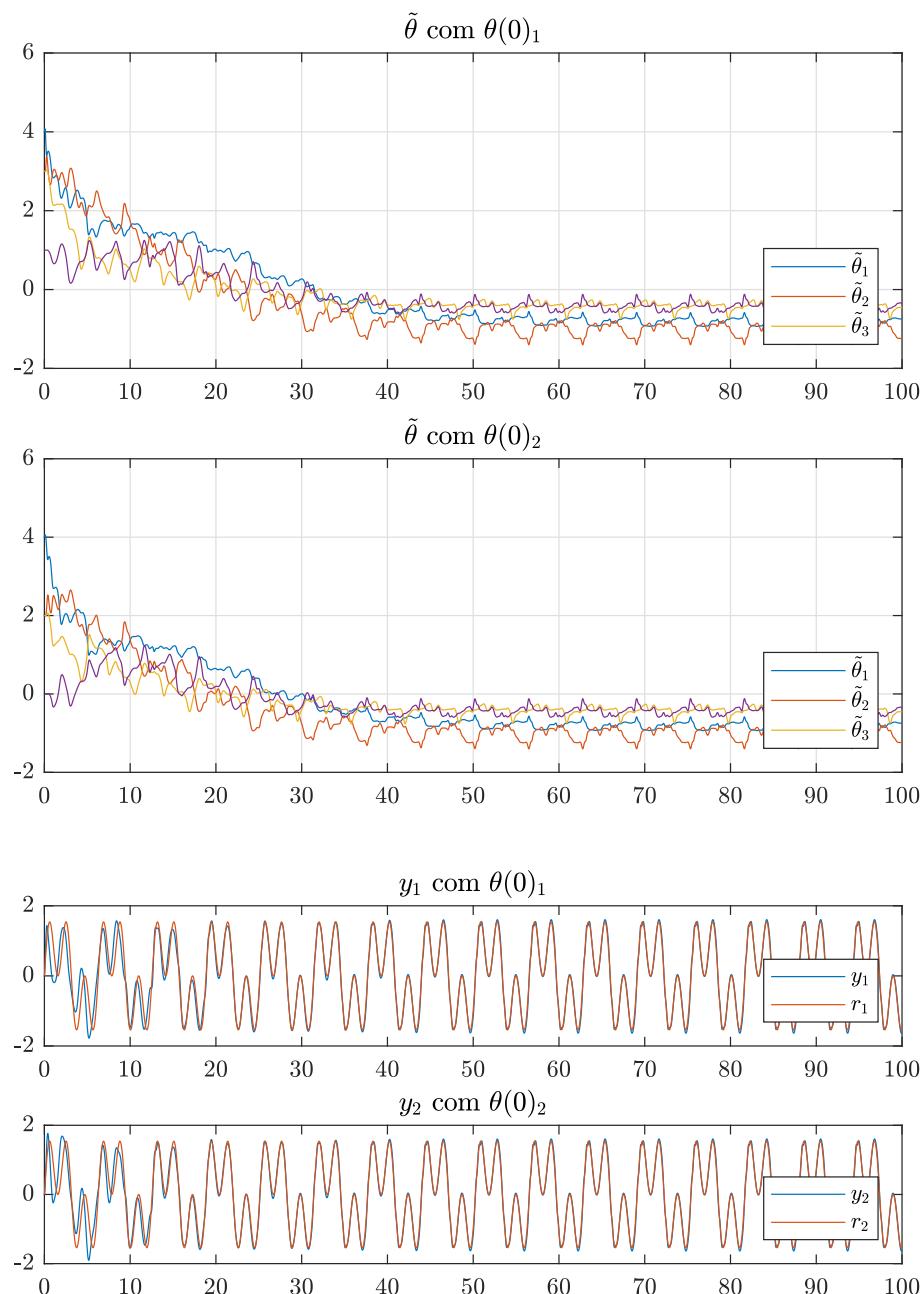
Inicialmente, desejamos verificar o comportamento do sistema para variações nas condições iniciais.

##### Simulação 1.1: $\theta(0)$

$$y = \frac{5}{s^3 + 3s^2 + 3s + 1} u, \quad \theta(0) = [0 \ 1], \quad y(0) = 0, \quad \Gamma = 0.5$$

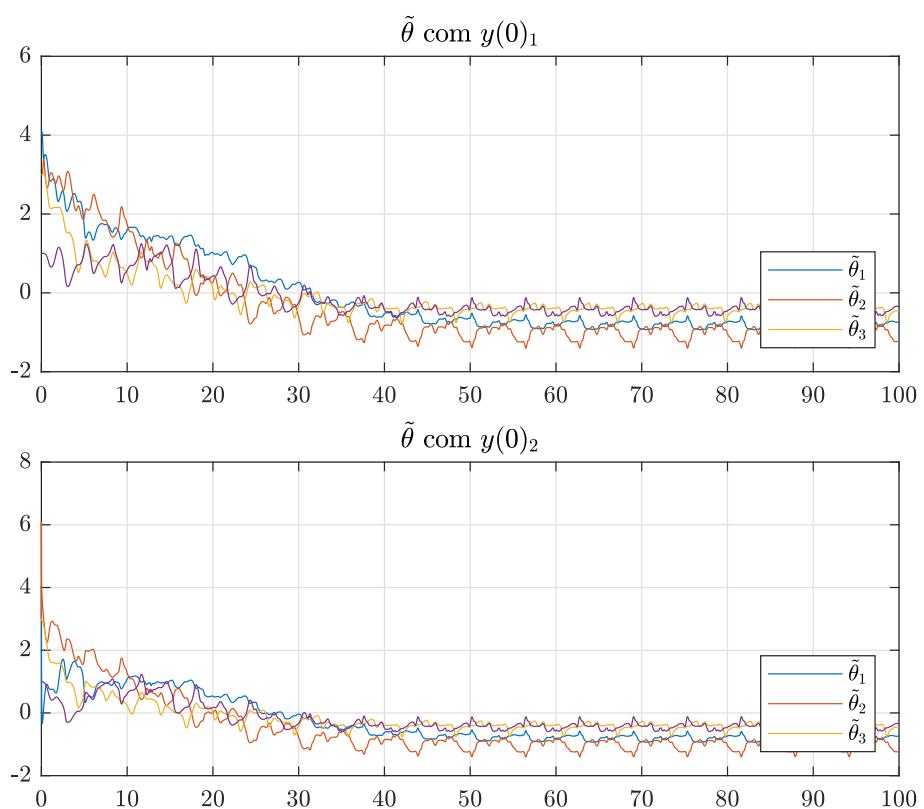
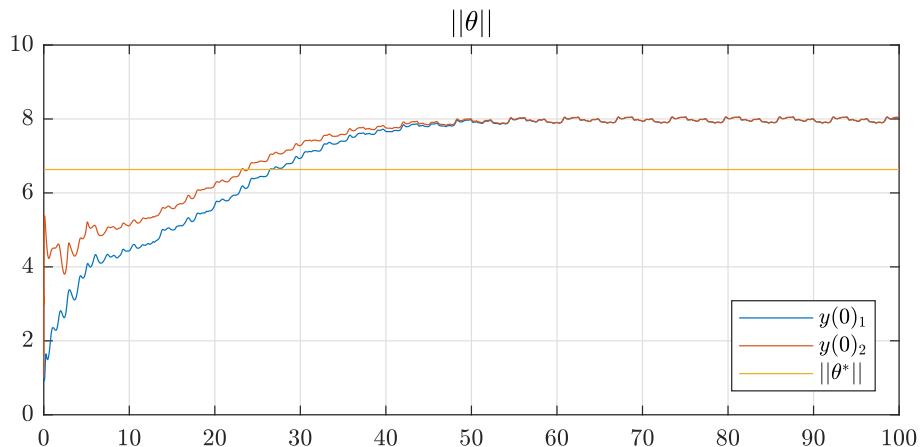
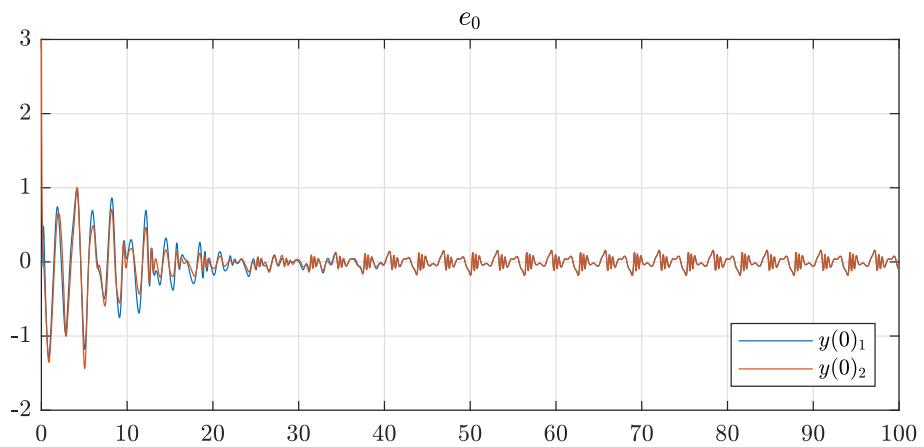
$$y_r = \sin(t) + \sin(3t) \quad k = [3 \ 3 \ 3].$$

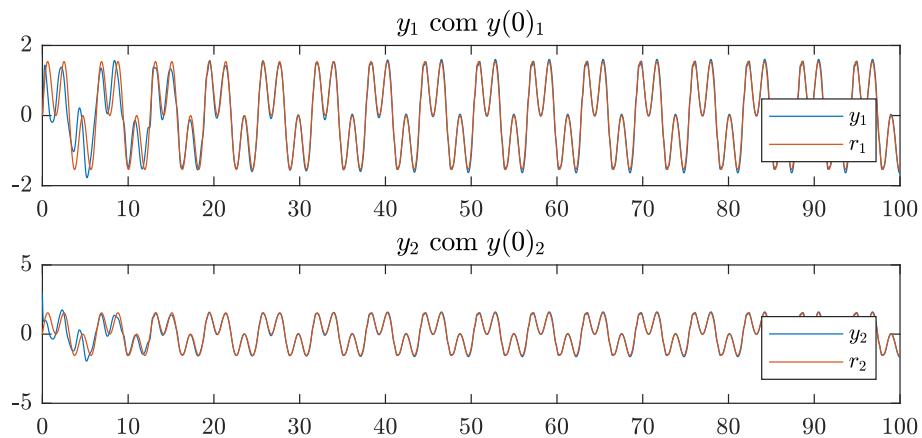


**Simulação 1.2:  $y(0)$** 

$$y = \frac{5}{s^3 + 3s^2 + 3s + 1} u, \quad \theta(0) = 0, \quad y(0) = [0 \ 3], \quad \Gamma = 0.5$$

$$y_r = \sin(t) + \sin(3t) \quad k = [3 \ 3 \ 3].$$



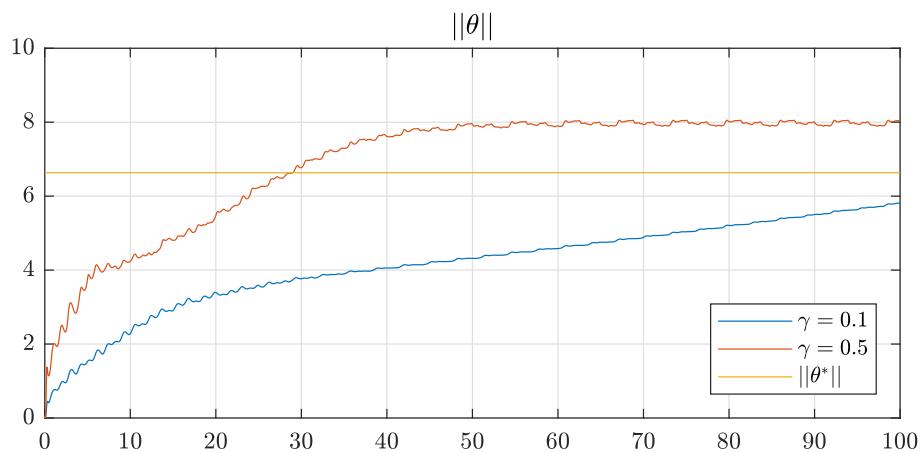
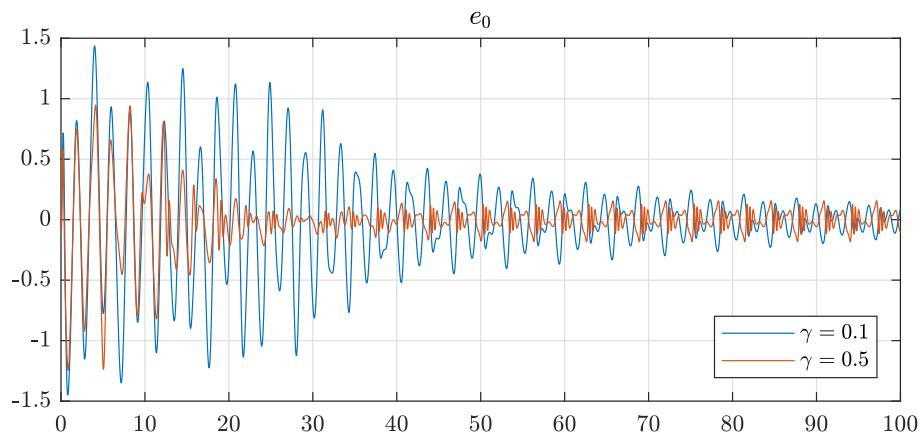


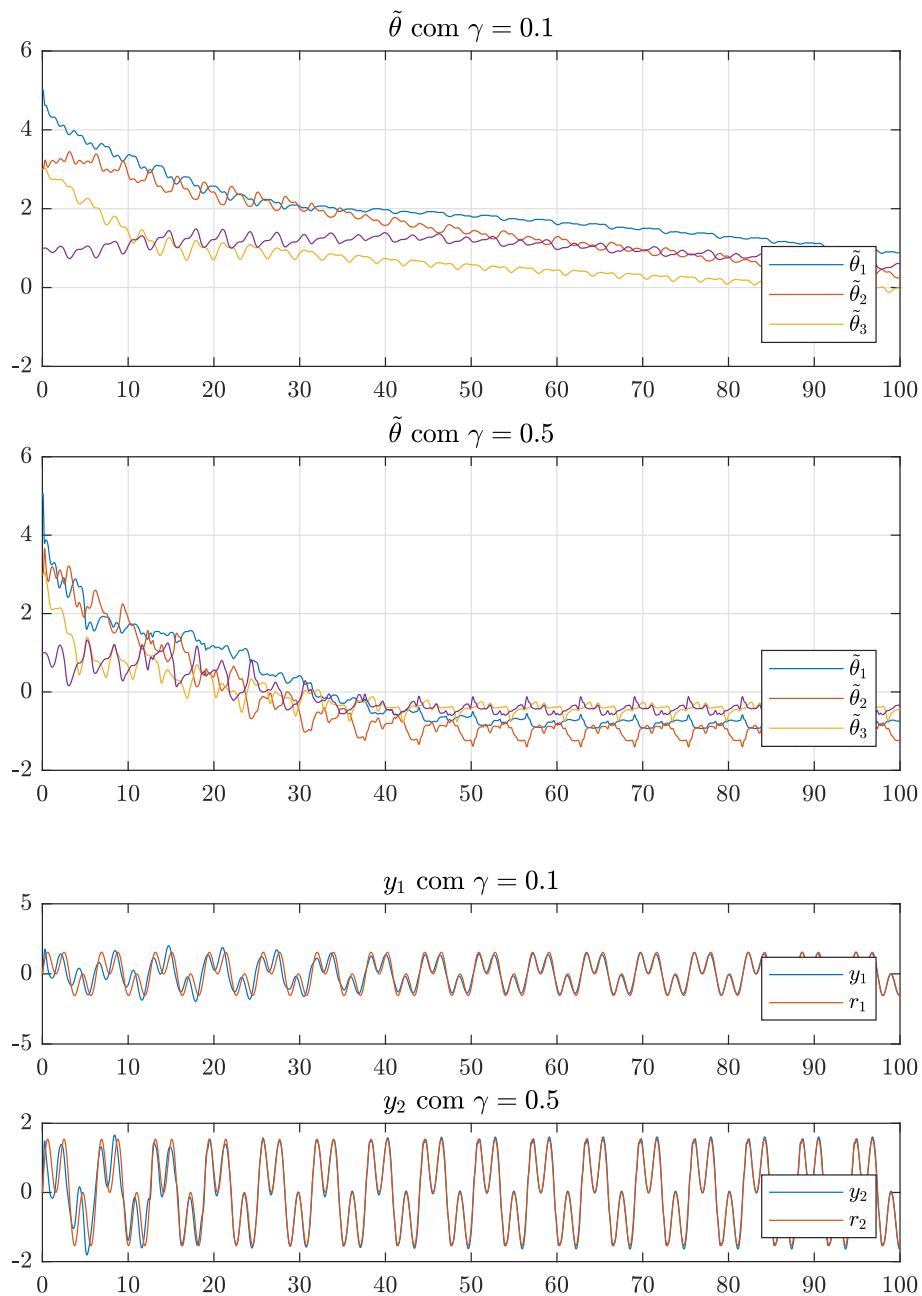
### 3.2 Simulação #2

Verificamos o comportamento do sistema para variações no parâmetro de adaptação  $\Gamma$ .

$$y = \frac{5}{s^3 + 3s^2 + 3s + 1} u, \quad \theta(0) = 0, \quad y(0) = 0, \quad \Gamma = [0.1 \ 0.5],$$

$$y_r = \sin(t) + \sin(3t) \quad k = [3 \ 3 \ 3].$$





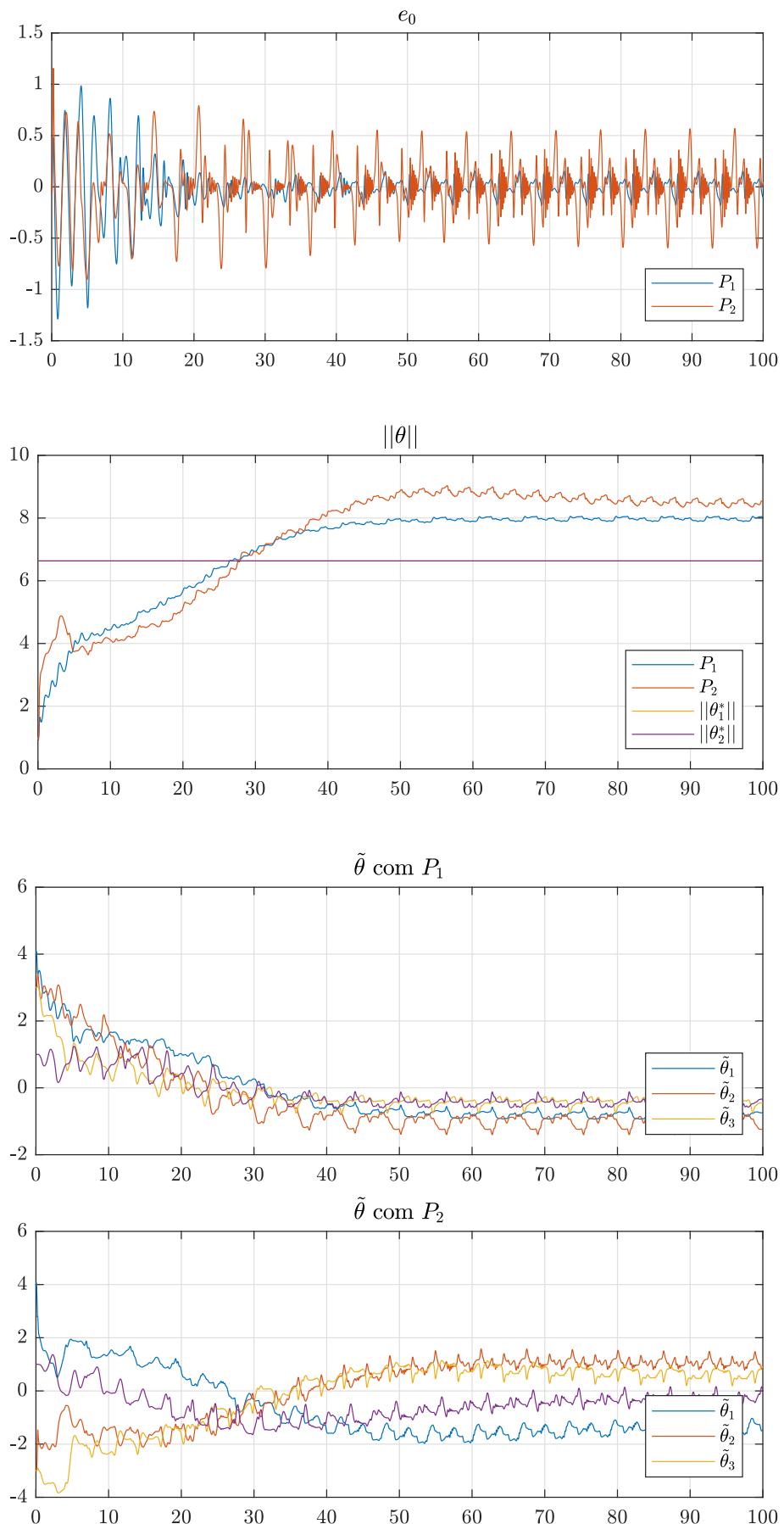
### 3.3 Simulação #3

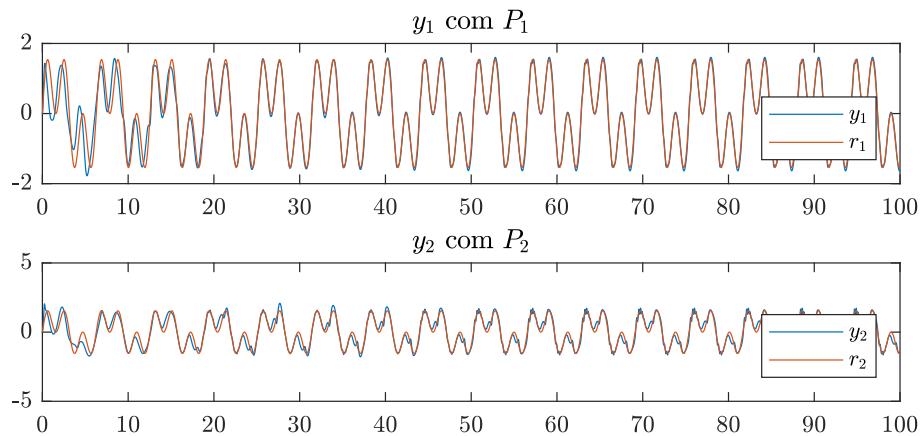
Verificamos o comportamento do sistema para variações na planta e modelo.

#### Simulação 3.1: planta

$$y = \frac{5}{s^3+3s^2+3s+1} u \text{ e } \frac{5}{s^3-3s^2-3s+1} u, \quad \theta(0) = 0, \quad y(0) = 0, \quad \Gamma = 0.5$$

$$y_r = \sin(t) + \sin(3t) \quad k = [3 \quad 3 \quad 3].$$

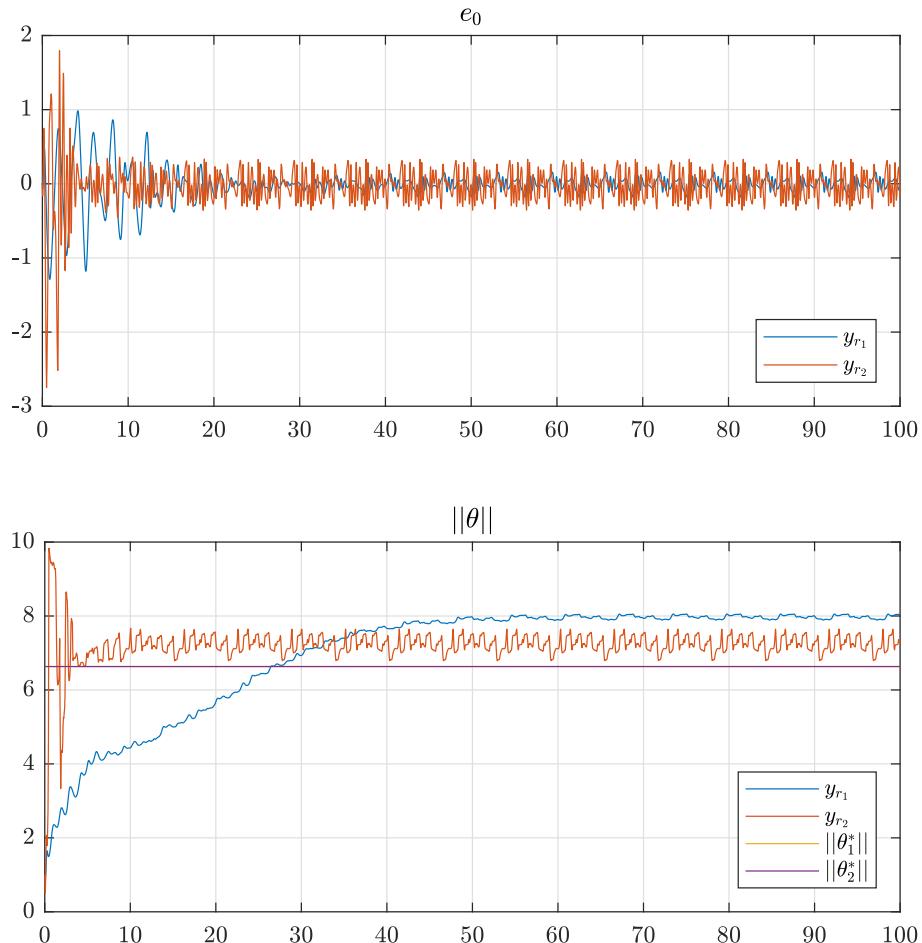


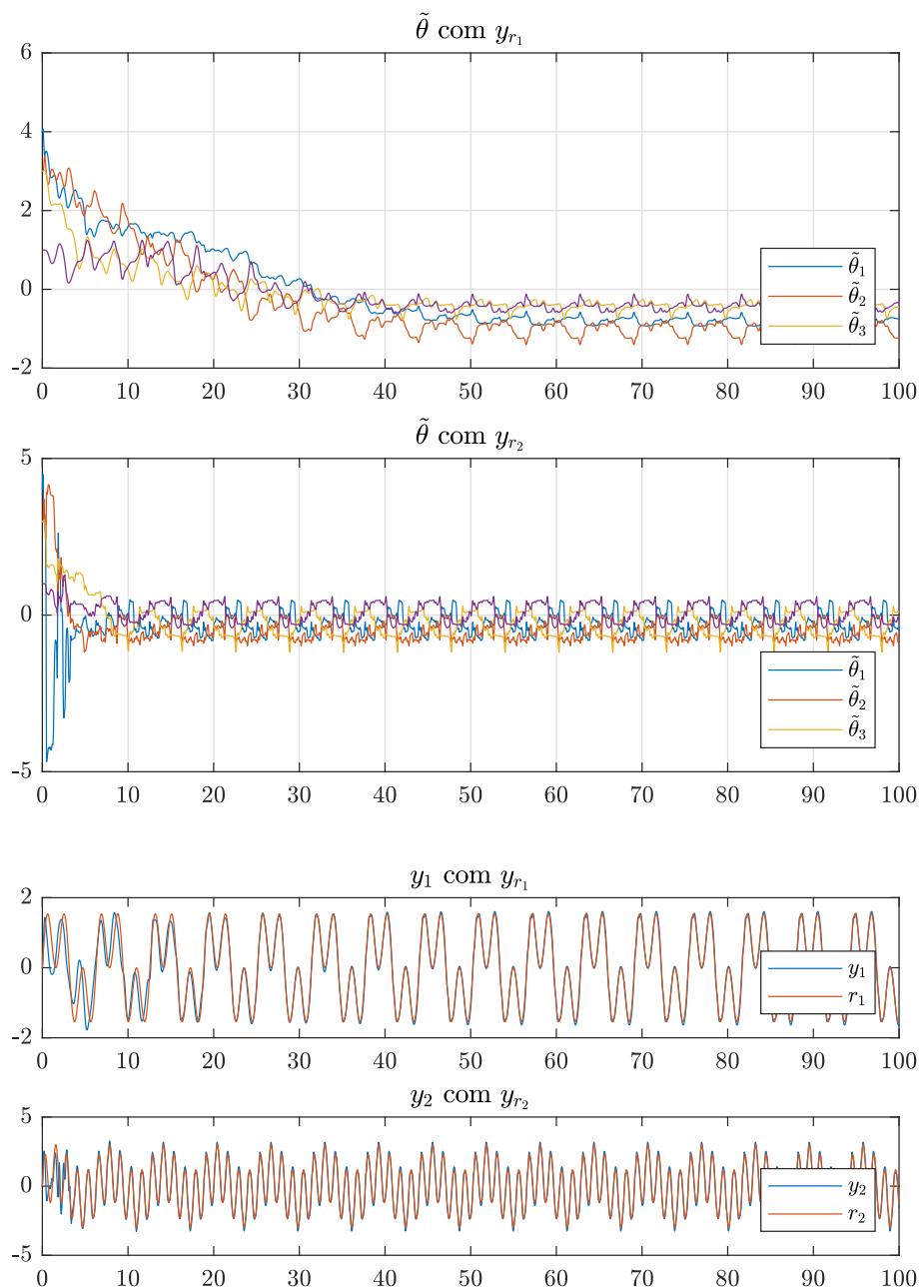


### Simulação 3.2: modelo

$$y = \frac{5}{s^3 + 3s^2 + 3s + 1} u \quad \theta(0) = 0, \quad y(0) = 0, \quad \Gamma = 0.5,$$

$y_r = [\sin(t) + \sin(3t)]$  e  $\sin(t) + 2\sin(5t)$ .





## 4 Discussão

Conforme aumentamos a ordem do Backstepping, as derivadas parciais crescem em grande complexidade, o que requer grande trabalho manual para otimizar o algoritmo. Além disso, o aumento do número de multiplicações e de novos termos impede que a simulação seja rápida. Verificou-se também a necessidade da diminuição do ganho de adaptação, se comparado com outros agoritmos como o MRAC, devido às multiplicações e possível overflow numérico.

A **simulação #1** testou o sistema para condições iniciais não nulas tanto dos parâmetros  $\theta$ , quanto do sistema  $y(0)$ . Observou-se a convergência do erro  $e_0$  para zero. Os parâmetros convergiram para valores próximos aos valores esperados.

A **simulação #2** testou o sistema para diferentes valores de ganho de adaptação:  $\Gamma = 0.1$  e  $\Gamma = 0.5$ . Para o maior valor,  $\Gamma = 0.5$ , o sistema convergiu mais rapidamente, com menor erro, porém com maiores oscilações. Observou-se overflow numérico para valores  $\Gamma > 10$ .

A **simulação #3.1** testou o algoritmo para diferentes sistemas, modelos de tranferência:  $Y/U = \frac{5}{s^3+3s^2+3s+1}$  e  $Y/U = \frac{5}{s^3-3s^2-3s+1}$ . Para a planta instável, o algoritmo leva muito tempo para convergir e as oscilações são maiores.

A **simulação #3.2** testou o algoritmo para diferentes entradas:  $r_1 = \sin(t) + \sin(3t)$  e  $r_2 = \sin(t) + 2\sin(5t)$ . Apesar de a entrada de alta frequência exigir mais do sistema, mostrando maior lentidão para a convergência, os parâmetros convergiram mais rapidamente.