Lista 2 - PTC-5719 Identificação de Sistemas

Mateus Chiqueto dos Santos

Maio 2025

1

Sabemos que os valores estimados para τ e t_s foram $\tau = 10s$ e $t_s = 40s$. Deste modo os valores para T_s serão: $T_s = \frac{10}{10} = 18s$ e $T_s = \frac{40}{10} = 4s$.

O melhor período de amostragem para ser escolhido é $1s$, pois permite amostrar aproximadamente 10 vezes durante a constante de tempo e cerca de 40 vezes durante o tempo de acomodação, ao contrário do outro valor em que isso se reduziria em cerca de 4 vezes.

Adiante utilizaremos $T_s = 1s$.

2

O processo é descrito por:

$$A(q)y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t);$$

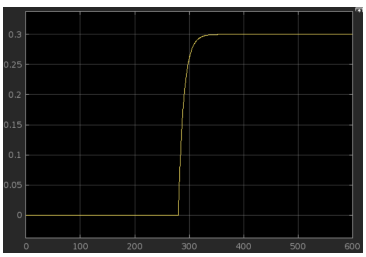
e os modelos e as ordens serão:

Estrutura	Polinômios Ativos	Ordem do Modelo
ARX	A, B	$n_a = 1, n_b = 1, n_k = 5$
ARMAX	A, B, C	$n_a = 1, n_b = 1, n_c = 2, n_k$
OE	B, F	$n_b = 1, n_f = 1, n_k = 5$
BJ	B, F, C, D	$n_b = 1, n_f = 1, n_c = 2, n_d = 2, n_k = 5$

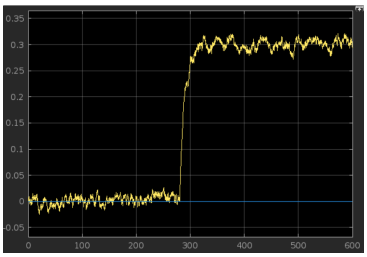
O BJ é mais indicado, pois pode capturar as perturbações filtradas de 1ª e 2ª ordem.

3

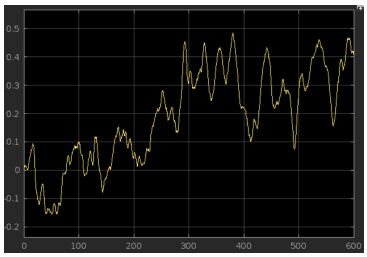
Seguem as simulações realizadas no simulink por $600s$.



Processo limpo



P. Baixa Intensidade



P. Alta Intensidade

4

No caso, como simularemos o processo limpo, as ordens serão $n_a = 1, n_b = 1, n_f = 1, n_c = 0, n_d = 0, n_k = 5, t_s = 42s$. Abaixo temos o código do *Matlab* para as aproximações

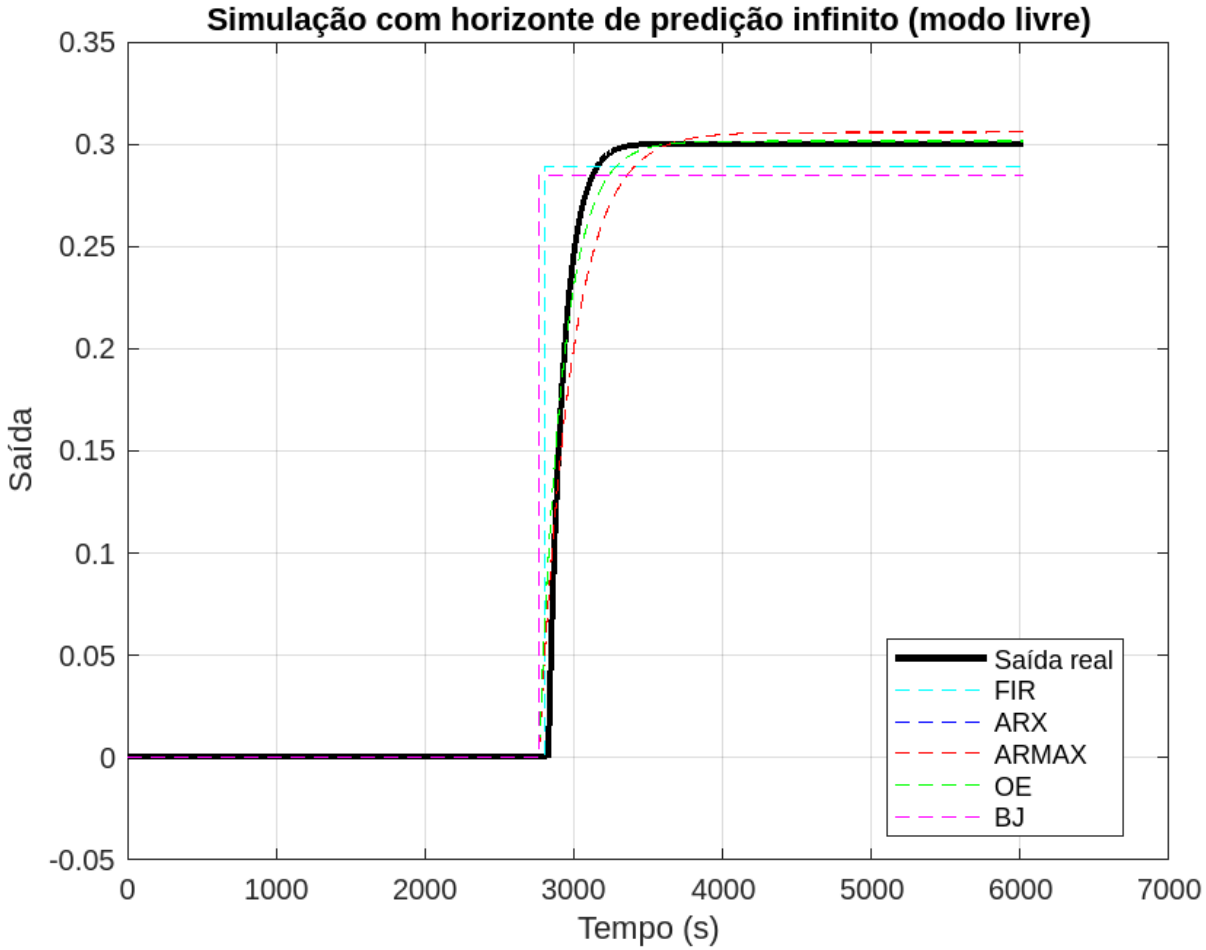
Listing 1: Identificação de modelos com dados limpos

```
%% 1. Prepara  o dos dados
y = out.branco;                    % Sa da limpa (sem ru do)
```

```
u = 0.1 * (out.tout >= 275);   % Entrada degrau de 0.1
Ts = 1;                        % Per odo de amostragem (1 s)
data = iddata(y, u, Ts);       % Objeto de identifica  o

%% 2. Par metros do sistema
nk = 5;                        % Atraso de tempo em amostras
ts_aprox = 42;                % Tempo de acomoda  o (ajustar conforme
    necess rio)

%% 3. Estima  o dos modelos

% FIR (modelo apenas com coeficientes de entrada)
modelo_fir = arx(data, [0 ts_aprox nk]);

% ARX
modelo_arx = arx(data, [1 1 nk]);

% ARMAX
modelo_armax = armax(data, [1 1 0 nk]);

% OE (Output Error)
modelo_oe = oe(data, [1 1 nk]);

% BJ (Box-Jenkins)
modelo_bj = bj(data, [1 1 0 0 nk]);

%% 4. Exibir os modelos e comparar com o modelo real (coloque a G(z) real aqui)
disp('--- Modelos Estimados ---');
disp('FIR:'); present(modelo_fir);
disp('ARX:'); present(modelo_arx);
disp('ARMAX:'); present(modelo_armax);
disp('OE:'); present(modelo_oe);
disp('BJ:'); present(modelo_bj);

% Exemplo para extrair os par metros
p_fir = getpvec(modelo_fir);
p_arx = getpvec(modelo_arx);
% Compare com os coeficientes da fun  o de transfer ncia discreta real do
    processo

%% 5. Simula  o com entrada degrau (modo livre, horizonte infinito)

% Simula  o usando sim() para prever sa da sem usar y real (pior caso)
y_sim_fir = sim(modelo_fir, u);
y_sim_arx = sim(modelo_arx, u);
y_sim_armax = sim(modelo_armax, u);
y_sim_oe = sim(modelo_oe, u);
y_sim_bj = sim(modelo_bj, u);

%% 6. Compara  o visual com resposta limpa (modo livre)
t = data.SamplingInstants;

figure;
plot(t, y, 'k', 'LineWidth', 2); hold on;
plot(t, y_sim_fir, '--c');
plot(t, y_sim_arx, '--b');
plot(t, y_sim_armax, '--r');
plot(t, y_sim_oe, '--g');
plot(t, y_sim_bj, '--m');
legend('Sa da real', 'FIR', 'ARX', 'ARMAX', 'OE', 'BJ');
title('Simula  o com horizonte de predi  o infinito (modo livre)');
xlabel('Tempo (s)');
ylabel('Sa da');
```

```
grid on;

%% 7. C lculo do  ndice  de ajuste (fit) para cada modelo

fit_fir = goodnessOfFit(y_sim_fir, y, 'NRMSE') * 100;
fit_arx = goodnessOfFit(y_sim_arx, y, 'NRMSE') * 100;
fit_armax = goodnessOfFit(y_sim_armax, y, 'NRMSE') * 100;
fit_oe = goodnessOfFit(y_sim_oe, y, 'NRMSE') * 100;
fit_bj = goodnessOfFit(y_sim_bj, y, 'NRMSE') * 100;

fprintf('\n---  ndice  de Ajuste (Fit %%) - Horizonte Infinito ---\n');
fprintf('FIR      : %.2f%%\n', fit_fir);
fprintf('ARX      : %.2f%%\n', fit_arx);
fprintf('ARMAX    : %.2f%%\n', fit_armax);
fprintf('OE       : %.2f%%\n', fit_oe);
fprintf('BJ       : %.2f%%\n', fit_bj);
```

Segue as curvas das aproximações:



Curvas referentes às aproximações e a simulação do processo limpo.

Veja os resultados na tabela 1:

Table 1: Índice de ajuste (Fit) dos modelos estimados

Modelo	Fit 1 passo (%)	Fit horizonte infinito (%)
FIR	79.31	20.69
ARX	**99.87**	8.10
ARMAX	**99.87**	8.10
OE	94.16	5.84
BJ	86.01	**26.21**

Observando os resultados notamos que ARX, ARMAX obtiveram os melhores resultados para 1 passo, mas valores baixos para infinito. E BJ se destacou obtendo os melhores resultados, pois foi mais consistente tanto em 1 passo tanto em infinitos passos.

5

Iremos calcular o ganho estacionário de cada aproximação utilizando os dados do exercício anterior com o código abaixo.

Listing 2: Cálculo do ganho estacionário

```
% Lista de modelos identificados
modelos = {
    'modelo_fir',
    'modelo_arx',
    'modelo_armax',
    'modelo_oe',
    'modelo_bj'
};

% Loop para calcular e mostrar o ganho DC de cada modelo
fprintf('--- Ganhos Estacionarios (DC Gain) ---\n');

for i = 1:length(modelos)
    nome = modelos{i};
    modelo = eval(nome);                    % Avalia o nome do modelo como vari vel
    [num, den] = tfdata(modelo, 'v');       % Extrai os polin mios B(z) e A(z)
    Kdc = sum(num) / sum(den);              % G(1) = B(1)/A(1)
    fprintf('%-12s : Kdc = %.6f\n', nome, Kdc);
end
```

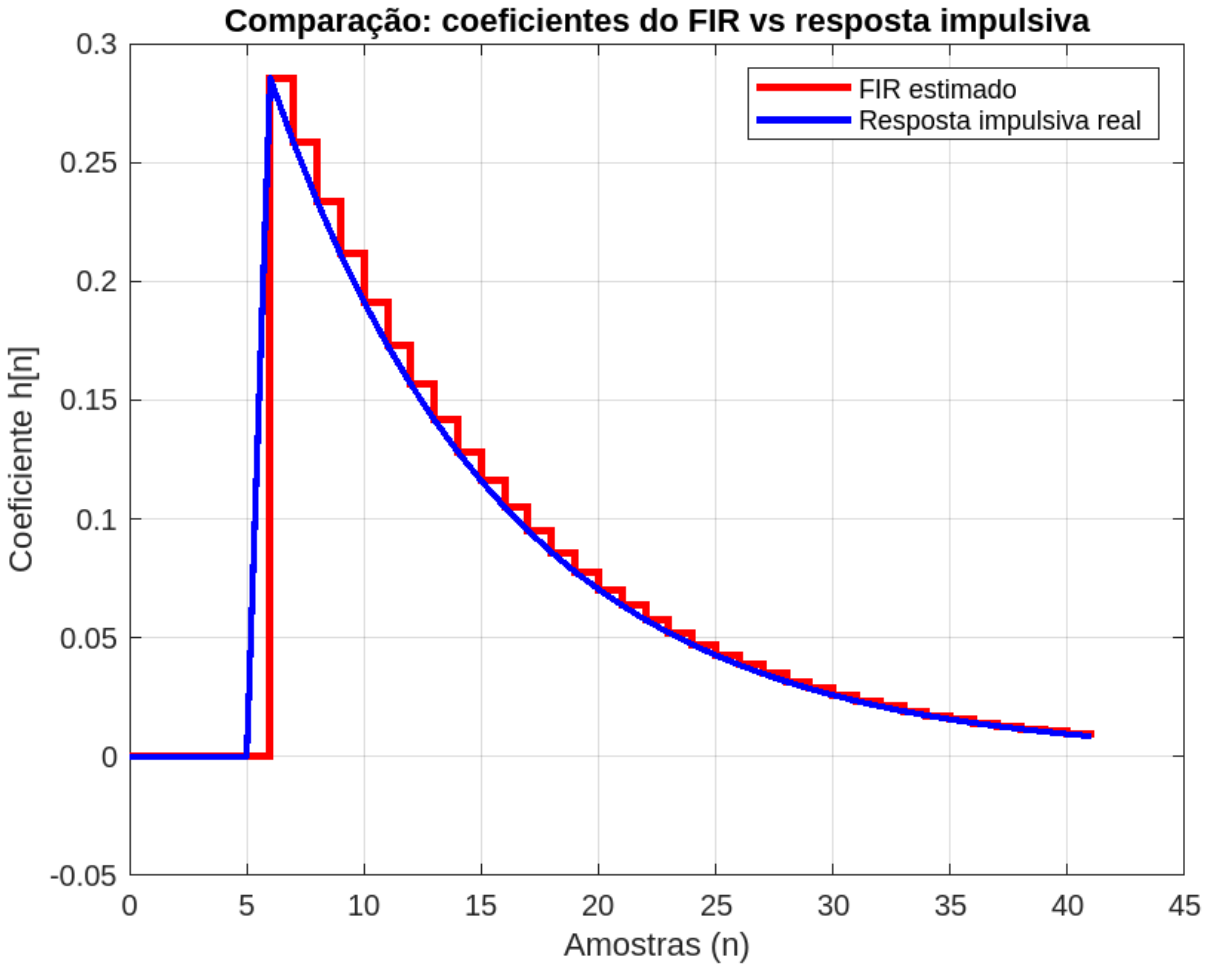
Os dados na tabela 2 são resultado da operação realizada pelo código acima. Na tabela é possível notar que os valores do ganho estacionário ficaram próximos a 3 que é o valor de K da simulação, a proximidade dos valores com 3 é guiada pelo fit de 1 passo visto no exercício anterior, onde quanto maior o fit, mais próximo do valor de 3 foi o ganho e quanto pior o fit de 1 passo, mais distante o valor.

Table 2: Ganho estacionário dos modelos estimados

Modelo	Ganho Estacionário
FIR	2.891365
ARX	3.058531
ARMAX	3.058531
OE	3.015694
BJ	2.848073

6

Na imagem abaixo vemos as curvas dos coeficientes FIR e a resposta impulsiva real. Veja que os coeficientes estão muito próximos da resposta real. Isso indica como o modelo fir é consistente.



Comparação entre os coeficientes FIR.

7

No código abaixo iremos realizar as aproximações para perturbações de baixa e alta intensidade.

Listing 3: Identificação de modelos com perturbacao

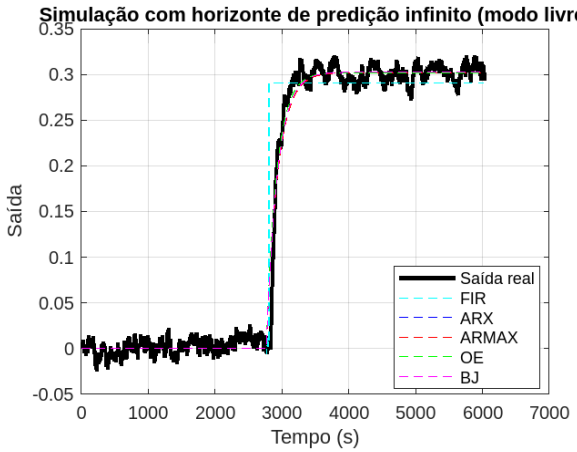
```
%%% Baixa intensidade
%% 1. Prepara  o dos dados
y = out.baixa;                  % Sa da limpa (sem ru do)
u = 0.1 * (out.tout >= 275);    % Entrada degrau de 0.1
Ts = 1;                         % Per odo de amostragem (1 s)
data = iddata(y, u, Ts);        % Objeto de identifica  o

%% 2. Par metros do sistema
nk = 5;                         % Atraso de tempo em amostras
ts_aprox = 42;                 % Tempo de acomoda  o (ajustar conforme
    necess rio)

%% 3. Estima  o dos modelos

% FIR (modelo apenas com coeficientes de entrada)
modelo_fir = arx(data, [0 ts_aprox nk]);
```

```
16
17 % ARX
18 modelo_arx = arx(data, [1 1 nk]);
19
20 % ARMAX
21 modelo_armax = armax(data, [1 1 1 nk]);
22
23 % OE (Output Error)
24 modelo_oe = oe(data, [1 1 nk]);
25
26 % BJ (Box-Jenkins)
27 modelo_bj = bj(data, [1 1 1 1 nk]);
28
29 %% 4. Exibir os modelos e comparar com o modelo real (coloque a G(z) real aqui)
30 disp('--- Modelos Estimados ---');
31 disp('FIR:'); present(modelo_fir);
32 disp('ARX:'); present(modelo_arx);
33 disp('ARMAX:'); present(modelo_armax);
34 disp('OE:'); present(modelo_oe);
35 disp('BJ:'); present(modelo_bj);
36
37 % Exemplo para extrair os par metros
38 p_fir = getpvec(modelo_fir);
39 p_arx = getpvec(modelo_arx);
40 % Compare com os coeficientes da fun  o de transfer ncia discreta real do
       processo
41
42 %% 5. Simula  o com entrada degrau (modo livre, horizonte infinito)
43
44 % Simula  o usando sim() para prever sa da sem usar y real (pior caso)
45 y_sim_fir = sim(modelo_fir, u);
46 y_sim_arx = sim(modelo_arx, u);
47 y_sim_armax = sim(modelo_armax, u);
48 y_sim_oe = sim(modelo_oe, u);
49 y_sim_bj = sim(modelo_bj, u);
50
51 %% 6. Compara  o visual com resposta limpa (modo livre)
52 t = data.SamplingInstants;
53
54 figure;
55 plot(t, y, 'k', 'LineWidth', 2); hold on;
56 plot(t, y_sim_fir, '--c');
57 plot(t, y_sim_arx, '--b');
58 plot(t, y_sim_armax, '--r');
59 plot(t, y_sim_oe, '--g');
60 plot(t, y_sim_bj, '--m');
61 legend('Sa da real', 'FIR', 'ARX', 'ARMAX', 'OE', 'BJ');
62 title('Simula  o com horizonte de predi  o infinito (modo livre)');
63 xlabel('Tempo (s)');
64 ylabel('Sa da');
65 grid on;
66
67 %% 7. C lculo do  ndice  de ajuste (fit) para cada modelo
68
69 fit_fir = goodnessOfFit(y_sim_fir, y, 'NRMSE') * 100;
70 fit_arx = goodnessOfFit(y_sim_arx, y, 'NRMSE') * 100;
71 fit_armax = goodnessOfFit(y_sim_armax, y, 'NRMSE') * 100;
72 fit_oe = goodnessOfFit(y_sim_oe, y, 'NRMSE') * 100;
73 fit_bj = goodnessOfFit(y_sim_bj, y, 'NRMSE') * 100;
74
75 fprintf('\n---  ndice  de Ajuste (Fit %%) - Horizonte Infinito ---\n');
76 fprintf('FIR     : %.2f%%\n', fit_fir);
77 fprintf('ARX     : %.2f%%\n', fit_arx);
```

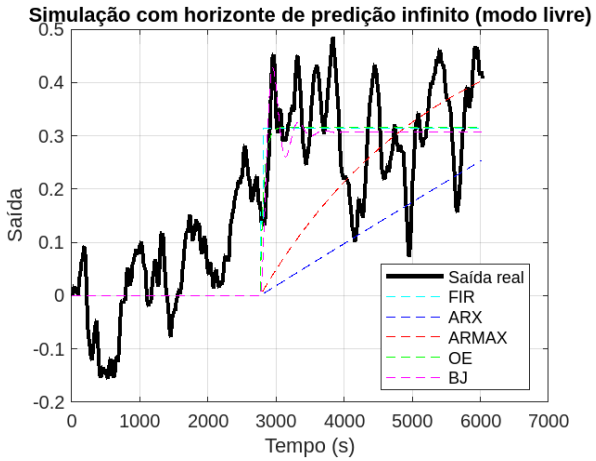
```
fprintf('ARMAX   : %.2f%%\n', fit_armax);
fprintf('OE      : %.2f%%\n', fit_oe);
fprintf('BJ      : %.2f%%\n', fit_bj);

%%% Alta intensidade
%% 1. Prepara  o dos dados
y = out.alta;                  % Sa da limpa (sem ru do)
u = 0.1 * (out.tout >= 275);    % Entrada degrau de 0.1
Ts = 1;                         % Per odo de amostragem (1 s)
data = iddata(y, u, Ts);        % Objeto de identifica  o

%% 2. Par metros do sistema
nk = 5;                         % Atraso de tempo em amostras
ts_aprox = 42;                 % Tempo de acomoda  o (ajustar conforme
    necess rio)

%% 3. Estima  o dos modelos

% FIR (modelo apenas com coeficientes de entrada)
modelo_fir = arx(data, [0 ts_aprox nk]);

% ARX
modelo_arx = arx(data, [1 1 nk]);

% ARMAX
modelo_armax = armax(data, [1 1 2 nk]);

% OE (Output Error)
modelo_oe = oe(data, [1 1 nk]);

% BJ (Box-Jenkins)
modelo_bj = bj(data, [1 1 2 2 nk]);

%% 4. Exibir os modelos e comparar com o modelo real (coloque a G(z) real aqui)
disp('--- Modelos Estimados ---');
disp('FIR:'); present(modelo_fir);
disp('ARX:'); present(modelo_arx);
disp('ARMAX:'); present(modelo_armax);
disp('OE:'); present(modelo_oe);
disp('BJ:'); present(modelo_bj);

% Exemplo para extrair os par metros
p_fir = getpvec(modelo_fir);
p_arx = getpvec(modelo_arx);
% Compare com os coeficientes da fun  o de transfer ncia discreta real do
    processo

%% 5. Simula  o com entrada degrau (modo livre, horizonte infinito)

% Simula  o usando sim() para prever sa da sem usar y real (pior caso)
y_sim_fir = sim(modelo_fir, u);
y_sim_arx = sim(modelo_arx, u);
y_sim_armax = sim(modelo_armax, u);
y_sim_oe = sim(modelo_oe, u);
y_sim_bj = sim(modelo_bj, u);

%% 6. Compara  o visual com resposta limpa (modo livre)
t = data.SamplingInstants;

figure;
plot(t, y, 'k', 'LineWidth', 2); hold on;
plot(t, y_sim_fir, '--c');
plot(t, y_sim_arx, '--b');
```

```
plot(t, y_sim_armax, '--r');
plot(t, y_sim_oe, '--g');
plot(t, y_sim_bj, '--m');
legend('Sa da real', 'FIR', 'ARX', 'ARMAX', 'OE', 'BJ');
title('Simula  o com horizonte de predi  o infinito (modo livre)');
xlabel('Tempo (s)');
ylabel('Sa da');
grid on;

%% 7. C lculo do  ndice  de ajuste (fit) para cada modelo

fit_fir = goodnessOfFit(y_sim_fir, y, 'NRMSE') * 100;
fit_arx = goodnessOfFit(y_sim_arx, y, 'NRMSE') * 100;
fit_armax = goodnessOfFit(y_sim_armax, y, 'NRMSE') * 100;
fit_oe = goodnessOfFit(y_sim_oe, y, 'NRMSE') * 100;
fit_bj = goodnessOfFit(y_sim_bj, y, 'NRMSE') * 100;

fprintf('\n---  ndice  de Ajuste (Fit %%) - Horizonte Infinito ---\n');
fprintf('FIR      : %.2f%%\n', fit_fir);
fprintf('ARX      : %.2f%%\n', fit_arx);
fprintf('ARMAX    : %.2f%%\n', fit_armax);
fprintf('OE       : %.2f%%\n', fit_oe);
fprintf('BJ       : %.2f%%\n', fit_bj);
```

Abaixo podemos observar as curvas das aproximações e as curvas reais. E na tabela 3 vemos os resultados



Aproximações ao modelo com p. de baixa intensidade.



Aproximações ao modelo com p. de alta intensidade.

de fit para 1 passo e para infinitos passos.

Table 3: Índice de ajuste (%) dos modelos identificados para diferentes intensidades

Modelo	**Intensidade**	**Fit 1 passo (%)**	**Fit infinito (%)**
FIR	Baixa	78.90	21.10
ARX	Baixa	98.65	8.48
ARMAX	Baixa	98.65	8.48
OE	Baixa	91.68	8.32
BJ	Baixa	98.65	8.39
FIR	Alta	37.81	62.19
ARX	Alta	99.10	106.29
ARMAX	Alta	99.72	84.60
OE	Alta	38.23	61.77
BJ	Alta	99.87	61.56

Na baixa intensidade, modelos como ARX, ARMAX e BJ apresentaram alto valor de fit em 1

passo, mas desempenho fraco para fit infinito, indicando que capturam apenas a resposta imediata, não a dinâmica completa. Já FIR e OE, apesar de menor ajuste em 1 passo, simularam melhor o comportamento do sistema. Na alta intensidade, ARX, ARMAX e BJ tiveram excelente desempenho tanto no ajuste em 1 passo quanto na simulação, refletindo boa capacidade de modelar a dinâmica do sistema ou o modelo está em sobre-ajuste.

8

Com código semelhante ao realizado no exercício 5 foi calculado os ganhos estacionários e são vistos na tabela 4.

Table 4: Ganhos Estacionários dos Modelos

Modelo	**Baixa Intensidade**	**Alta Intensidade**
FIR	2.9029	3.1362
ARX	3.0210	62.5678
ARMAX	3.0210	5.6499
OE	3.0215	3.1645
BJ	3.0252	3.0709

Veja que os modelos OE e BJ foram os mais consistentes nas aproximações para as duas simulações, mantendo os ganhos próximos ao real. E o modelo ARX foi instável pois funcionou bem para baixa intensidade, porém teve resultado ruim para alta intensidade.

9

Abaixo temos o código de Matlab para as comparações agora com o medidor:

Listing 4: Estimando com medidor

```
%% 1. Par metros do sistema
Ts = 1;              % Per odo de amostragem
N = 600;             % N mero de amostras (600 s)
t = (0:N-1)' * Ts;   % Vetor de tempo

% Entrada: degrau a partir de t = 275 s
u = double(t >= 275) * 0.1;

%% 2. Gerar e1 com ru do e perturba  es
rng(1); % para reprodutibilidade
e1 = sqrt(0.001) * randn(N, 1);               % ru do de excita  o
e1_medida = e1 + sqrt(1e-6) * randn(N, 1);    % e1 medida com ru do

% Perturba  o v1: e1 passado por FT de 1   ordem: Gv1(z)
Gv1 = tf(1, [5 1]);           % cont nua
Gv1d = c2d(Gv1, Ts);          % discreta com ZOH
v1 = lsim(Gv1d, e1, t);

% Perturba  o v2: outro ru do filtrado (nova semente)
rng(2);
e2 = sqrt(0.001) * randn(N,1);
Gv2 = tf(2, conv([5 1], [10 1]));
Gv2d = c2d(Gv2, Ts);
v2 = lsim(Gv2d, e2, t);

%% 3. Gerar ru do de medi  o
y_noise = sqrt(1e-6) * randn(N, 1);

%% 4. Sistema principal: G(s)     G(z)
Gp = tf(3, [10 1]);          % sem atraso
```

```
Gd = c2d(Gp, Ts, 'zoh');      % discretizado
% Incluir atraso de 5s      5 amostras (em Ts = 1s)
Gd.InputDelay = 5;

y_sistema = lsim(Gd, u, t);

%% 5. Compor a sa da com perturba  es e ru do
y_baixa = y_sistema + v1 + v2 + y_noise;

%% 6. Preparar dados para identifica  o com 2 entradas
data_2in = iddata(y_baixa, [u e1_medida], Ts);

% Para compara  o: identifica  o com 1 entrada (s  u)
data_1in = iddata(y_baixa, u, Ts);

%% 7. Estimar modelos (exemplo com ARX)
na = 2; nb = [2 2]; nk = [1 1];  % para duas entradas
modelo_2in = arx(data_2in, [na nb nk]);

nb1 = 2; nk1 = 1; % para entrada  nica
modelo_1in = arx(data_1in, [na nb1 nk1]);

%% 8. Comparar os dois modelos
figure;
compare(data_2in, modelo_2in, modelo_1in);
legend('Dados reais', 'Modelo com 2 entradas', 'Modelo com 1 entrada');
title('Compara  o: modelo com e sem perturba  o medida');

%% 9. Avaliar  ndices  fit
[~, fit2in, ~] = compare(data_2in, modelo_2in);
[~, fit1in, ~] = compare(data_2in, modelo_1in);
fprintf('Fit com 2 entradas: %.2f%%\n', fit2in);
fprintf('Fit com 1 entrada: %.2f%%\n', fit1in);
```

Deste modo, veja na imagem 1 as curvas e os valores de fit. Veja que o modelo com o medidor obteve melhor resultado em comparação ao com 1 entrada. O modelo com duas entradas é mais preciso porque modela uma fonte adicional de variação da saída, que o modelo com uma entrada não consegue capturar.

10

Abaixo temos o código utilizado para medir os ganhos estacionários.

Listing 5: Ganho estacionário dos modelos estimados

```
K_1in = dcgain(modelo_1in);     % Modelo com 1 entrada (u)
K_2in = dcgain(modelo_2in(1)); % Modelo com 2 entradas (apenas entrada u)
```

Deste modo, tivemos:
- Modelo com 1 entrada: K = 2.9718 (Erro = 0.94%)
- Modelo com 2 entradas: K = 2.9954 (Erro = 0.15%).

Veja que, o modelo com 2 entradas apresenta um erro de estimativa menor que o modelo com 1 entrada. Isso confirma que medir e incluir a perturbação mais relevante (e1) melhora a qualidade da identificação do sistema, tornando a estimativa do ganho estacionário mais precisa.

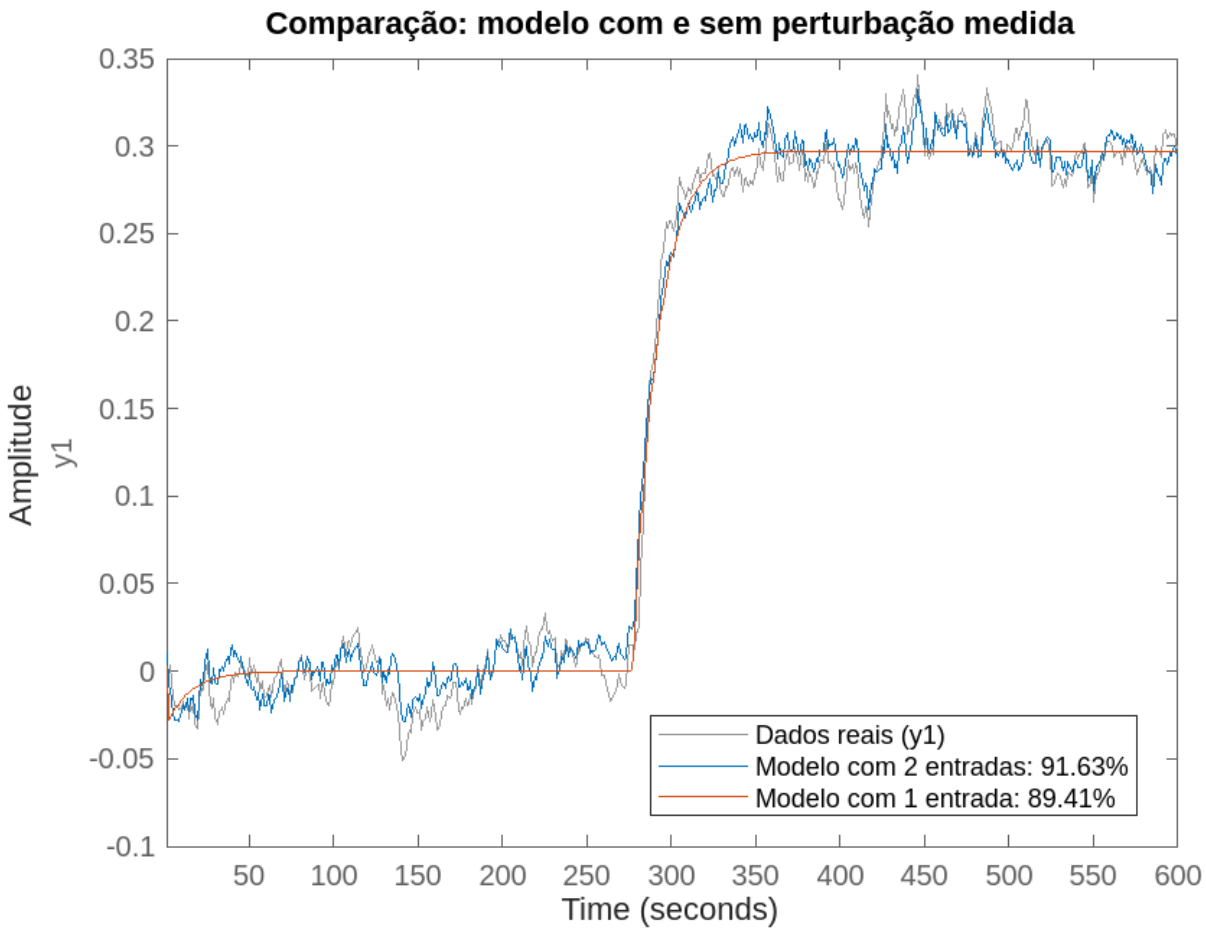


Figure 1:
Comparação entre as aproximações.