Lista 3 - PTC-5719 Identificação de Sistemas

Mateus Chiqueto dos Santos

Junho 2025

1 a

Para plotar os sinais utilizaremos o código abaixo.

Listing 1: Geração e comparação de sinais PRBS

```
N = 1001;
Amplitude = 0.1;
T_amostragem = 1;
Tb_lento = 20 * T_amostragem;
u_prbs_lento_raw = idinput(N, 'prbs', [0 1/(Tb_lento/T_amostragem)], [-1 1]);
sinal_prbs_lento = u_prbs_lento_raw * Amplitude;
Tb_rapido = 4 * T_amostragem;
u_prbs_rapido_raw = idinput(N, 'prbs', [0 1/(Tb_rapido/T_amostragem)], [-1 1]);
sinal_prbs_rapido = u_prbs_rapido_raw * Amplitude;
tempo = (0:N-1) * T_amostragem;
figure;
stairs(tempo, sinal_prbs_lento, 'b', 'DisplayName', 'PRBS Lento (T_b = 20s)');
hold on;
stairs(tempo, sinal_prbs_rapido, 'r--', 'DisplayName', 'PRBS R pido (T_b = 4s)
    ');
title('Compara  o de Sinais PRBS: Lento vs. R pido');
xlabel('Tempo (s)');
ylabel('Amplitude');
ylim([-0.12 0.12]);
grid on;
legend('show');
hold off;
```

Segue na figura 1 os sinais plotados no mesmo gráfico. Pela figura 1 podemos notar que o PRBS rápido gera muitos mais sinais do que o lento, mesmo com a mesma amplitude, o PRBS rápido pode captar comportamento mais complexo do processo por gerar mais dados em pequeno espaço de tempo.

2 b

Para gerarmos a resposta do processo ao PRBS Lento utilizaremos somente o Matlab, segue o código abaixo.

Listing 2: Resposta ao processo com PRBS lento

```
K = 3;
tau = 10; % s
theta = 5; % s
T_amostragem = 1;
Ts = T_amostragem;
K_pert1 = 1;
tau_pert1 = 5;
K_pert2 = 2;
tau_pert2_1 = 5;
tau_pert2_2 = 10;
variancia_baixa = 0.001;
variancia_alta = 0.1;
```

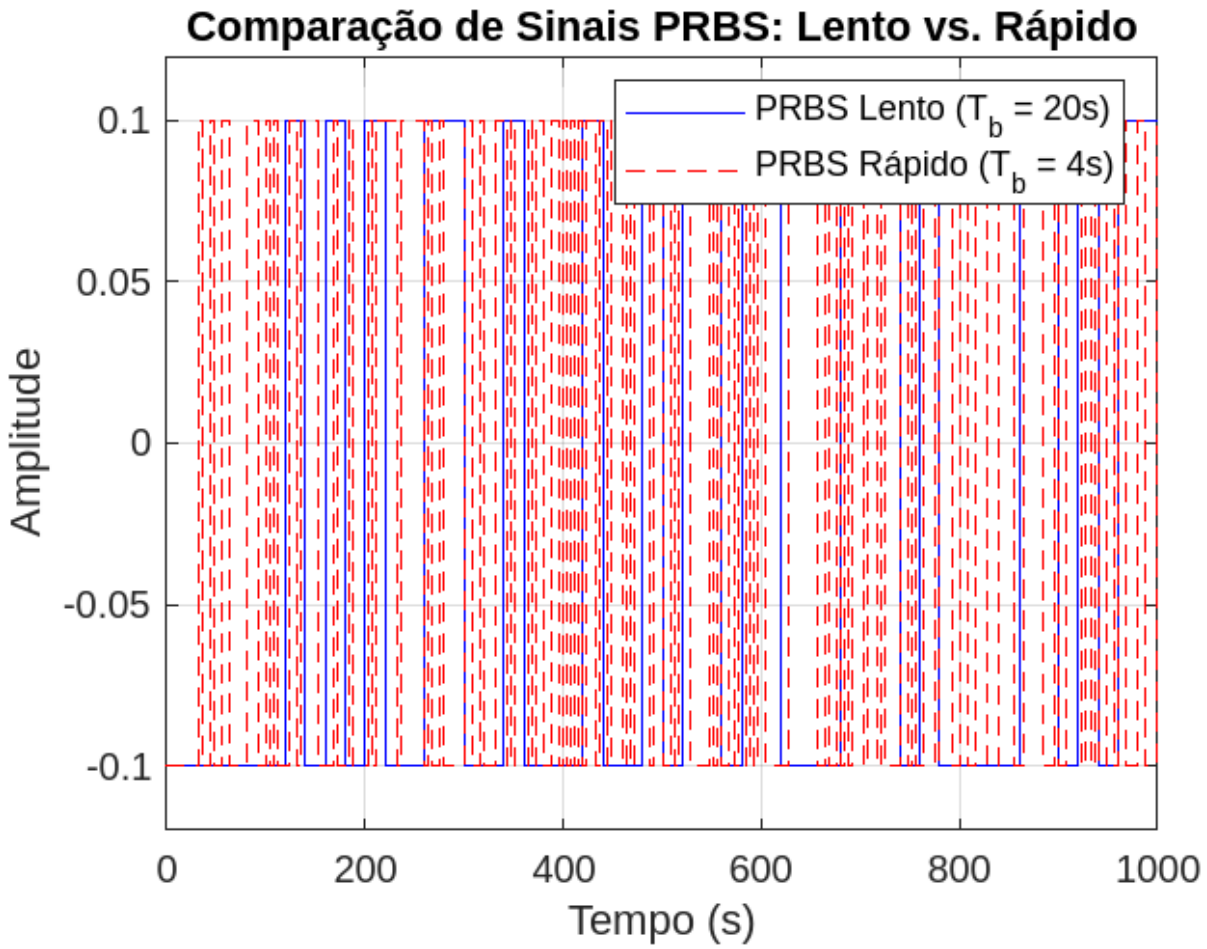

Figure 1: Comparação de sinais PRBS lento e rápido.

```
variancia_ruido_medicao = 1e-6;
tempo_simulacao = 1000;
num_pontos = tempo_simulacao / Ts + 1;
tempo = (0:Ts:tempo_simulacao)';
Amplitude_prbs = 0.1;
Tb_lento_Ex1 = 20 * Ts;
u_prbs_lento_raw = idinput(num_pontos, 'prbs', [0 1/(Tb_lento_Ex1/Ts)], [-1 1])
    ;
u_prbs_lento = u_prbs_lento_raw * Amplitude_prbs;
sys_c = tf(K, [tau 1], 'InputDelay', theta);
sys_d = c2d(sys_c, Ts, 'zoh');
[num_d, den_d] = tfdata(sys_d, 'v');
sys_pert1_c = tf(K_pert1, [tau_pert1 1]);
sys_pert1_d = c2d(sys_pert1_c, Ts, 'zoh');
[num_pert1_d, den_pert1_d] = tfdata(sys_pert1_d, 'v');
den_pert2_c = conv([tau_pert2_1 1], [tau_pert2_2 1]);
sys_pert2_c = tf(K_pert2, den_pert2_c);
sys_pert2_d = c2d(sys_pert2_c, Ts, 'zoh');
[num_pert2_d, den_pert2_d] = tfdata(sys_pert2_d, 'v');
rng(1001);
e_v1_baixa_seq = randn(num_pontos, 1) * sqrt(variancia_baixa);
rng(1002);
e_v2_baixa_seq = randn(num_pontos, 1) * sqrt(variancia_baixa);
rng(1003);
e_medicao_baixa_seq = randn(num_pontos, 1) * sqrt(variancia_ruido_medicao);
rng(2001);
e_v1_alta_seq = randn(num_pontos, 1) * sqrt(variancia_alta);
```

```
rng(2002);
e_v2_alta_seq = randn(num_pontos, 1) * sqrt(variancia_alta);
rng(2003);
e_medicao_alta_seq = randn(num_pontos, 1) * sqrt(variancia_ruido_medicao);
y_limpa = zeros(num_pontos, 1);
y_baixa = zeros(num_pontos, 1);
y_alta = zeros(num_pontos, 1);
states_proc_limpa = zeros(max(length(num_d), length(den_d)) - 1, 1);
states_proc_baixa_int = zeros(max(length(num_d), length(den_d)) - 1, 1);
states_proc_alta_int = zeros(max(length(num_d), length(den_d)) - 1, 1);
states_pert1_baixa = zeros(max(length(num_pert1_d), length(den_pert1_d)) - 1,
    1);
states_pert2_baixa = zeros(max(length(num_pert2_d), length(den_pert2_d)) - 1,
    1);
states_pert1_alta = zeros(max(length(num_pert1_d), length(den_pert1_d)) - 1, 1)
    ;
states_pert2_alta = zeros(max(length(num_pert2_d), length(den_pert2_d)) - 1, 1)
    ;
for i = 1:num_pontos
    u_curr = u_prbs_lento(i);
    [y_limpa_curr, states_proc_limpa] = filter(num_d, den_d, u_curr,
        states_proc_limpa);
    y_limpa(i) = y_limpa_curr;
    [v1_baixa_curr, states_pert1_baixa] = filter(num_pert1_d, den_pert1_d,
        e_v1_baixa_seq(i), states_pert1_baixa);
    [v2_baixa_curr, states_pert2_baixa] = filter(num_pert2_d, den_pert2_d,
        e_v2_baixa_seq(i), states_pert2_baixa);
    y_baixa(i) = y_limpa(i) + v1_baixa_curr + v2_baixa_curr +
        e_medicao_baixa_seq(i)
    [v1_alta_curr, states_pert1_alta] = filter(num_pert1_d, den_pert1_d,
        e_v1_alta_seq(i), states_pert1_alta);
    [v2_alta_curr, states_pert2_alta] = filter(num_pert2_d, den_pert2_d,
        e_v2_alta_seq(i), states_pert2_alta);
    y_alta(i) = y_limpa(i) + v1_alta_curr + v2_alta_curr + e_medicao_alta_seq(i
        );
end
figure;
plot(tempo, y_limpa, 'b', 'DisplayName', 'Sa da Y Limpa');
hold on; % Permite adicionar m ltiplos plots    mesma figura
plot(tempo, y_baixa, 'r', 'DisplayName', 'Sa da Y com Baixa Perturba  o');
plot(tempo, y_alta, 'g', 'DisplayName', 'Sa da Y com Alta Perturba  o');
title('Resposta do Processo ao PRBS Lento');
xlabel('Tempo (s)');
ylabel('Sa da Y');
legend('show');
grid on;
hold off;
```

Após rodar a simulação temos o gráfico com as respostas na figura 2.

3 c

Os resultados da validação cruzada está na tabela 1.

4 d

Os resultados da validação cruzada para PRBS rápido estão na tabela 2. Podemos notar que os resultados pioraram, modelos identificados com excitação degrau tendem a validar melhor com sinais de excitação mais lentos e podemos concluir que ruídos intensos prejudicam o ajuste.

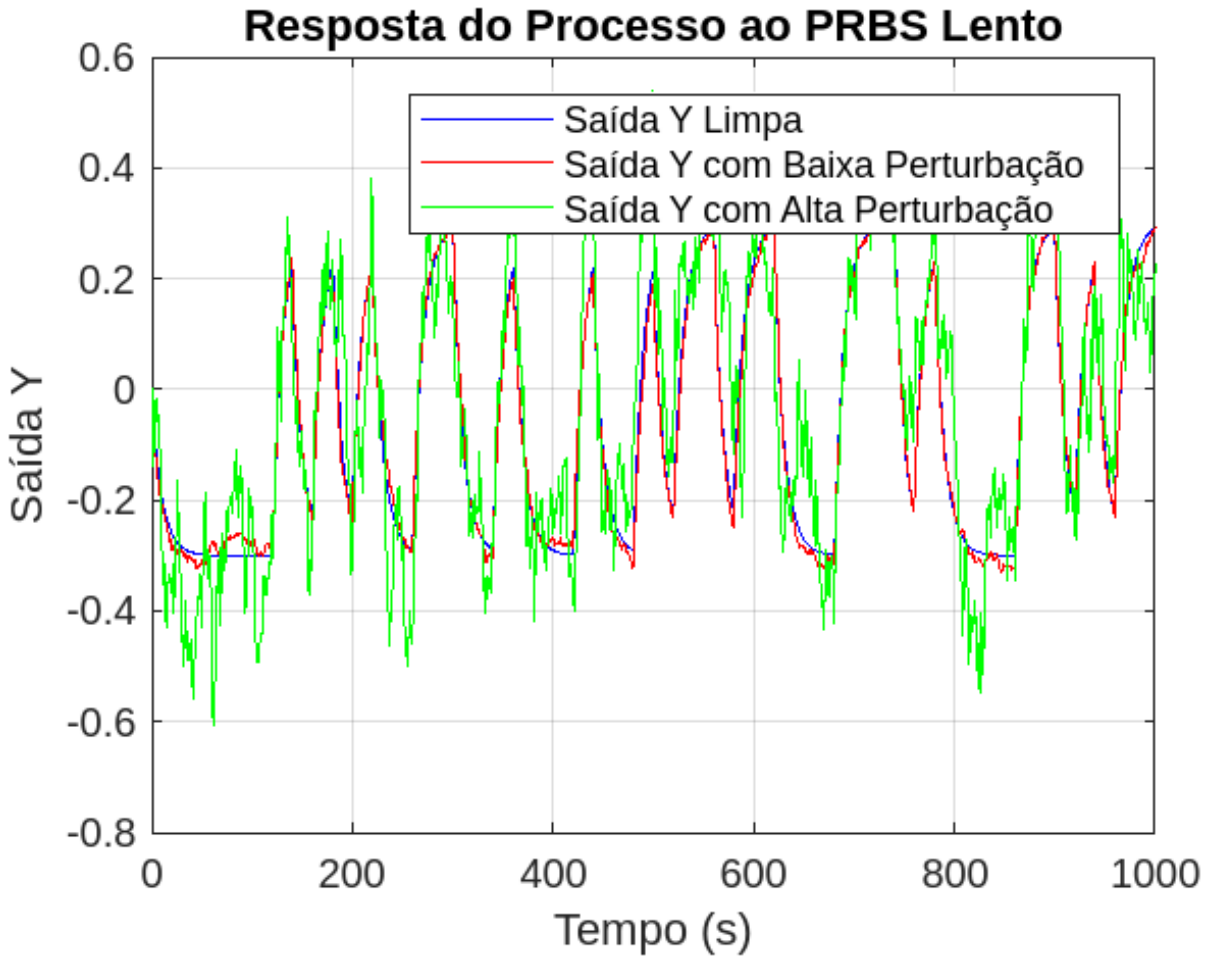

Figure 2: Resposta do processo ao PRBS lento.

Table 1: Resultados dos Valores de Fit (PRBS Lento)

Modelo	Saída Limpa	Perturbação Baixa	Perturbação Alta
FIR	-0.2564	-0.2179	-0.0671
ARX	76.6297	68.6780	7.6102
ARMAX	85.7920	74.4995	7.8724
OE	89.2854	76.3374	7.8798
BJ	89.0361	76.2129	7.9131

Table 2: Resultados dos Valores de Fit (PRBS Rápido)

Modelo	Saída Limpa	Perturbação Baixa	Perturbação Alta
FIR	1.8356	1.0116	-0.0259
ARX	-18.2354	-10.9673	-0.3665
ARMAX	-15.4365	-9.6248	-0.4172
OE	-28.2880	-18.2022	-0.7196
BJ	-23.0156	-14.6094	-0.5862

5 e

Neste caso o código de Matlab abaixo gera as simulações e na tabela 3 e 4 temos os modelos gerados para PRBS lento e rápido, respectivamente.

Listing 3: Código para aproximação ao processo com PRBS lento

```
na = 1; nb = 1; nf = 1; nc = 2; nd = 2; nk = 5;

% ARX
model_arx_baixa   = arx(data_baixa,   [na nb nk]);
model_arx_alta    = arx(data_alta,    [na nb nk]);
% ARMAX
model_armax_baixa = armax(data_baixa, [na nb nc nk]);
model_armax_alta  = armax(data_alta,  [na nb nc nk]);
% OE
model_oe_baixa    = oe(data_baixa,    [nb nf nk]);
model_oe_alta     = oe(data_alta,     [nb nf nk]);
% BJ
model_bj_baixa    = bj(data_baixa,    [nb nc nd nf nk]);
model_bj_alta     = bj(data_alta,     [nb nc nd nf nk]);
```

Table 3: Modelos de Identificação para Baixa Intensidade

Modelo	Função de Transferência	Ajuste (Fit)	FPE
ARX	$\frac{B(z)}{A(z)} = \frac{0.2739z^{-5}}{1-0.8523z^{-1}}$	59.27%	0.003765
ARMAX	$\frac{B(z)}{A(z)} = \frac{0.2573z^{-5}}{1-0.9337z^{-1}}$ *com ruído* $\frac{C(z)}{A(z)} = \frac{1-0.8648z^{-1}-0.04987z^{-2}}{1-0.9337z^{-1}}$	68.64%	0.00224
OE	$\frac{B(z)}{F(z)} = \frac{0.2573z^{-5}}{1-0.933z^{-1}}$	68.53%	0.00224
BJ	$\frac{B(z)}{F(z)} = \frac{0.2574z^{-5}}{1-0.9338z^{-1}}$ *com ruído* $\frac{C(z)}{D(z)} = \frac{1-0.7685z^{-1}-0.1743z^{-2}}{1-0.8364z^{-1}-0.1253z^{-2}}$	68.65%	0.002253

Table 4: Modelos de Identificação para Alta Intensidade

Modelo	Função de Transferência	Ajuste (Fit)	FPE
ARX	$\frac{B(z)}{A(z)} = \frac{0.3894z^{-5}}{1-0.05211z^{-1}}$	0.3029%	0.2146
ARMAX	$\frac{B(z)}{A(z)} = \frac{0.2401z^{-5}}{1-0.9404z^{-1}}$ *com ruído* $\frac{C(z)}{A(z)} = \frac{1-0.9716z^{-1}+0.02168z^{-2}}{1-0.9404z^{-1}}$	4.224%	0.1987
OE	$\frac{B(z)}{F(z)} = \frac{0.2403z^{-5}}{1-0.9409z^{-1}}$	4.147%	0.1977
BJ	$\frac{B(z)}{F(z)} = \frac{0.3223z^{-5}}{1+0.8411z^{-1}}$ *com ruído* $\frac{C(z)}{D(z)} = \frac{1+0.2438z^{-1}-0.6528z^{-2}}{1+0.1851z^{-1}-0.7364z^{-2}}$	0.7211%	0.215

6 f

Segue abaixo o código para realização da validação cruzada com os 400 pontos finais. A partir disso temos na tabela 5 os resultados e nota-se que a aproximação FIR obteve o melhor resultado para ambos casos, lento e rápido. Igualmente nos exercícios anteriores a aproximação à resposta do processo ao PRBS alto tem qualidade menor que o lento.

Listing 4: Validação Cruzada aos 400 pontos finais

```
N_val = 400;
u_val = u_prbs(end-N_val+1:end);
y_baixa_val = y_baixa(end-N_val+1:end);
y_alta_val  = y_alta(end-N_val+1:end);
data_baixa_val = iddata(y_baixa_val, u_val, T);
```

```
data_alta_val  = iddata(y_alta_val,  u_val, T);
[~, fit_baixa_fir]   = compare(data_baixa_val, model_fir_baixa);
[~, fit_baixa_arx]   = compare(data_baixa_val, model_arx_baixa);
[~, fit_baixa_armax] = compare(data_baixa_val, model_armax_baixa);
[~, fit_baixa_oe]    = compare(data_baixa_val, model_oe_baixa);
[~, fit_baixa_bj]    = compare(data_baixa_val, model_bj_baixa);
[~, fit_alta_fir]   = compare(data_alta_val, model_fir_alta);
[~, fit_alta_arx]   = compare(data_alta_val, model_arx_alta);
[~, fit_alta_armax] = compare(data_alta_val, model_armax_alta);
[~, fit_alta_oe]    = compare(data_alta_val, model_oe_alta);
[~, fit_alta_bj]    = compare(data_alta_val, model_bj_alta);
fit_table = table( ...
    [fit_baixa_fir;   fit_alta_fir], ...
    [fit_baixa_arx;   fit_alta_arx], ...
    [fit_baixa_armax; fit_alta_armax], ...
    [fit_baixa_oe;    fit_alta_oe], ...
    [fit_baixa_bj;    fit_alta_bj], ...
    'VariableNames', {'FIR','ARX','ARMAX','OE','BJ'}, ...
    'RowNames', {'Baixa','Alta'} ...
);
disp(' ndices  fit (%) para valida  o cruzada nos 400 pontos finais:');
disp(fit_table);
```

Table 5: Índices fit (%) para validação cruzada nos 400 pontos finais

	FIR	ARX	ARMAX	OE	BJ
Baixa	70.686	52.698	68.407	68.234	68.562
Alta	7.1226	-0.35981	5.9753	5.9681	6.2351

7 g

Para este caso, abaixo temos o código de matlab que realiza os cálculos e na tabela6 os índices fit.

Listing 5: Validação Cruzada ao degrau

```
T = 1;                % Intervalo de amostragem
N_deg = 600;          % Dura  o da simula  o (600 s)
u_deg = zeros(N_deg,1);
u_deg(276:end) = 0.1; % Degrau de 0.1 em t=275s ( ndice  276)
t_deg = (0:N_deg-1)*T;
K = 3; tau = 10; theta = 5;
s = tf('s');
G = K*exp(-theta*s)/(tau*s + 1);
Gd = c2d(G, T, 'zoh');
% Sementes iguais  s  dos itens anteriores
rng(10); v1_baixa = sqrt(0.001)*randn(N_deg,1);
rng(20); v2_baixa = sqrt(0.001)*randn(N_deg,1);
rng(30); v1_alta = sqrt(0.1)*randn(N_deg,1);
rng(40); v2_alta = sqrt(0.1)*randn(N_deg,1);
rng(100); e_med = sqrt(1e-6)*randn(N_deg,1);
% Sa da limpa (sem perturba  o nem ru do)
y_limpo = lsim(Gd, u_deg, t_deg);
% Sa da com perturba  es de baixa intensidade
y_baixa = lsim(Gd, u_deg, t_deg) + v1_baixa + v2_baixa + e_med;
% Sa da com perturba  es de alta intensidade
y_alta  = lsim(Gd, u_deg, t_deg) + v1_alta  + v2_alta  + e_med;
data_baixa_deg = iddata(y_baixa, u_deg, T);
data_alta_deg  = iddata(y_alta,  u_deg, T);
% Valida  o para perturba  o baixa
[~, fit_baixa_fir]   = compare(data_baixa_deg, model_fir_baixa);
```

```
[~, fit_baixa_arx]   = compare(data_baixa_deg, model_arx_baixa);
[~, fit_baixa_armax] = compare(data_baixa_deg, model_armax_baixa);
[~, fit_baixa_oe]    = compare(data_baixa_deg, model_oe_baixa);
[~, fit_baixa_bj]    = compare(data_baixa_deg, model_bj_baixa);
% Valida  o para perturba  o alta
[~, fit_alta_fir]   = compare(data_alta_deg, model_fir_alta);
[~, fit_alta_arx]   = compare(data_alta_deg, model_arx_alta);
[~, fit_alta_armax] = compare(data_alta_deg, model_armax_alta);
[~, fit_alta_oe]    = compare(data_alta_deg, model_oe_alta);
[~, fit_alta_bj]    = compare(data_alta_deg, model_bj_alta);
fit_table = table( ...
    [fit_baixa_fir;   fit_alta_fir], ...
    [fit_baixa_arx;   fit_alta_arx], ...
    [fit_baixa_armax; fit_alta_armax], ...
    [fit_baixa_oe;    fit_alta_oe], ...
    [fit_baixa_bj;    fit_alta_bj], ...
    'VariableNames', {'FIR','ARX','ARMAX','OE','BJ'}, ...
    'RowNames', {'Baixa','Alta'} ...
)
```

Table 6: Índices fit (%) para validação cruzada ao degrau

	FIR	ARX	ARMAX	OE	BJ
Baixa	38.64	45.153	43.055	43.357	43.071
Alta	5.9538	0.82607	3.8291	3.8207	4.1778

8 h

Para calcular e comparar os ganhos estacionários utilizaremos o código de Matlab abaixo. O ganho real do processo é $K = 3$. Após rodar o código os resultados estão na tabela 7.

Podemos notar pelos resultados que as aproximações OE, BJ e ARMAX obtiveram ganhos mais próximos ao real em abos os casos, lento e rápido, muito provavelmente por modelarem o ruído.

Listing 6: Ganhos estacionários

```
getGain = @(model) dcgain(model); % Usa fun  o dcgain do MATLAB
gain_fir_baixa   = getGain(model_fir_baixa);
gain_arx_baixa   = getGain(model_arx_baixa);
gain_armax_baixa = getGain(model_armax_baixa);
gain_oe_baixa    = getGain(model_oe_baixa);
gain_bj_baixa    = getGain(model_bj_baixa);
gain_fir_alta    = getGain(model_fir_alta);
gain_arx_alta    = getGain(model_arx_alta);
gain_armax_alta  = getGain(model_armax_alta);
gain_oe_alta     = getGain(model_oe_alta);
gain_bj_alta     = getGain(model_bj_alta);

gains_baixa = [gain_fir_baixa; gain_arx_baixa; gain_armax_baixa; gain_oe_baixa;
     gain_bj_baixa];
gains_alta  = [gain_fir_alta; gain_arx_alta; gain_armax_alta; gain_oe_alta;
    gain_bj_alta];
modelos = {'FIR'; 'ARX'; 'ARMAX'; 'OE'; 'BJ'};
T_ganho = table(gains_baixa, gains_alta, ...
    'RowNames', modelos, ...
    'VariableNames', {'Ganho_Baixa', 'Ganho_Alta'});
disp('Ganho estacion rio dos modelos:');
disp(T_ganho);
fprintf('Ganho real do processo: %.4f\n', K_real);
```

Table 7: Ganho estacionário dos modelos

	Ganho_Baixa	Ganho_Alta
FIR	1.8053	2.4531
ARX	1.9765	0.85635
ARMAX	4.0228	3.9401
OE	4.0149	3.9436
BJ	4.0223	3.9618

9 i

Para realizar a validação cruzada porém com PRBS rápido iremos utilizar os mesmos códigos anteriores, porém alterado para PRBS rápido e não lento. Os resultados dos cálculos estão na tabela 8.

Comparando os resultados da tabela 8, referente ao PRBS rápido, e da tabela 6, referente ao PRBS lento, vemos que a aproximação utilizando PRBS lento é mais efetiva, pois todos índices fit foram negativos no caso rápido. Prováveis motivos podem ser pela resposta ao degrau depende de baixas frequências e também por altas frequências não capturarem a dinâmica do modelo.

Table 8: Índices fit (%) para validação cruzada ao degrau (modelos PRBS rápido)

	FIR	ARX	ARMAX	OE	BJ
Baixa	-63.82	-46.644	-48.417	-49.741	-47.696
Alta	-2.7046	-7.6856	-7.1626	-7.4187	-7.418

10 j

Para o cálculo dos ganhos estacionários o código é semelhante ao utilizado no exercício "i", alterando apenas o PRBS para rápido. Na tabela 9.

Mais uma vez, observando os ganhos estacionários na tabela 7 observamos que os modelos OE e BJ apresentaram os melhores resultados, principalmente no processo com PRBS lento. Com PRBS rápido o modelo tende a variar mais, logo os melhores resultados foram com PRBS lento.

Table 9: Ganho estacionário dos modelos (PRBS rápido)

	Rapido_Baixa	Rapido_Alta
FIR	-0.57903	-1.8981
ARX	-0.027989	0.35081
ARMAX	7.5639	0.35787
OE	-0.11955	0.35416
BJ	3.553	0.52081