TRABALHO PRÁTICO - Grupo 13

Laboratório de Sistemas Dinâmicos
Professor: Bruno Barbosa

Kethellen Camilo Martins- 201720616 - Turma C
Michelli de Oliveira Bento - 201710884 - Turma A
Rafael Barbosa Souza - 201820522 - Turma A
Vinícius Faria Silveira - 201720627 - Turma C

Lavras-MG
Abril de 2021

Sumário

1 Introdução

Sabe-se que na maioria dos problemas reais, o modelo do sistema não é conhecido. O que se tem, entretanto, são apenas os sinais de entrada e saída. Assim sendo, métodos determinísticos para a modelagem caixa preta, podem ser empregados a fim de obtermos a resposta ao impulso do sistema. Esta prática foi desenvolvida para aplicações dos métodos caixa preta sendo utilizado o software MATLAB.

2 Geração de dados

Para realização do ensaio foi utilizado as seguintes variáveis:

- Número do grupo: 13;
- Período de Amostragem para entrada degrau : 0.1s.
- Período de Amostragem para entrada PRBS: 0.18s.

A partir desses valores foi obtido a seguinte figura pelo LabView:

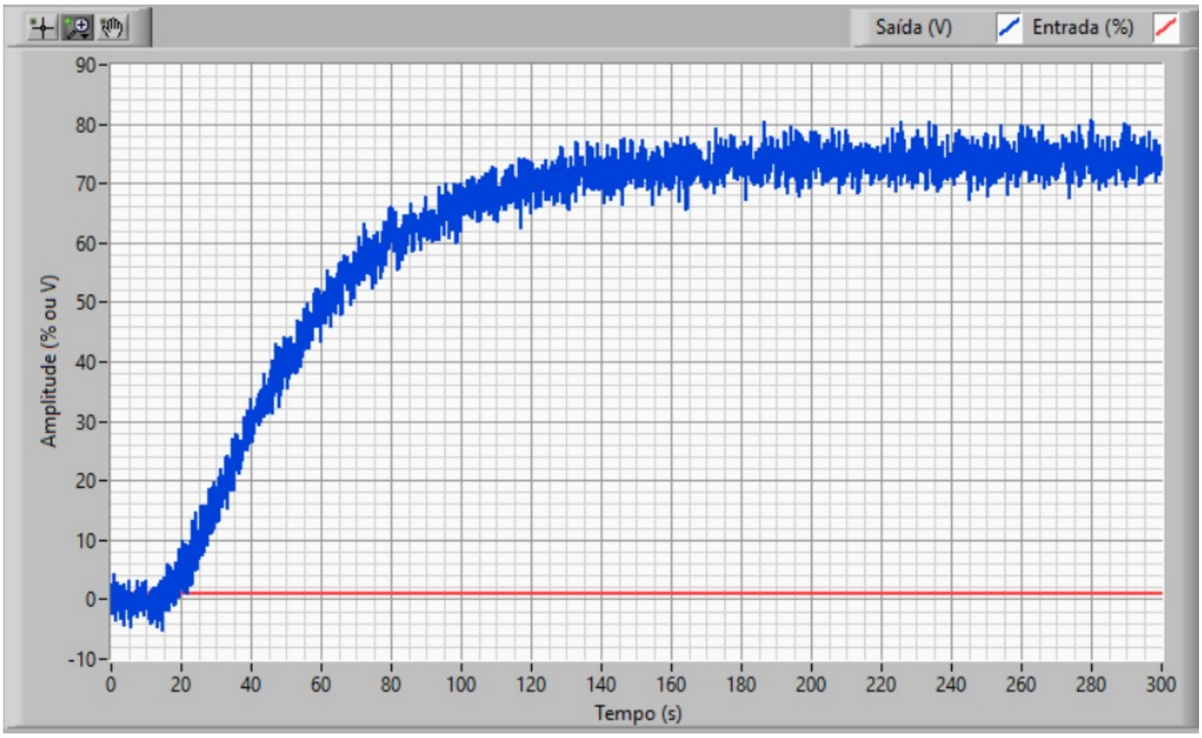

Figura 1: *Resposta do sistema para uma entrada degrau unitário.*

De acordo com a figura acima temos em vermelho a entrada do degrau unitário e em azul a saída medida de um processo real. A partir desses dados foi gerado um arquivo .txt que será utilizado no software MATLAB para análise dos métodos.

3 Modelos de Primeira Ordem

A partir da equação 1 aplicável à sistemas de primeira ordem com atraso puro de tempo mostrada abaixo, podemos desenvolver os seguintes métodos:

- Método da integrais;
- Método de uma constante de tempo (63,2%);
- Método de quatro constantes de tempo (98%);
- Método de Ziegler- Nichols;
- Método de Smith;
- Método de Haglund;
- Método de Sundaresan e Krishnaswami;
- Método da inclinação inicial.

$$H(s) = \frac{Ke^{-\theta s}}{\tau s + 1} \tag{1}$$

Temos que:

- K: Ganho do sistema;
- τ: Constante de tempo;
- θ: Tempo morto.

3.1 Método da Integrais

Para aplicar o método da integrais foi utilizado o seguinte código no MATLAB:

```
% Metodo de integrais%
%%
dados_degrau = load('saidaLab.txt');
t = dados_degrau(:,1);
u = dados_degrau(:,2);
y = dados_degrau(:,3);
%%
%Percebemos que a saida tende ao seu valor permanente a partir do tempo
%220s, cujo  ndice    1100.

dy = mean(y(1100:3000)) - mean(y(1:86));
du = u(end)-0;
K = dy/du;
```

```

%% Normalizando o vetor
yN = y/K;

%%
teta_tal = trapz(t,u-yN);
indice=find(t <=teta_tal);
i =585;
tal = exp(1)*trapz(t(1:i),yN(1:i));
teta = teta_tal - tal;

%% Funcao de transferencia

s = tf('s');
G1 = (K*exp(-teta*s))/(tal*s +1)
step(G1, '-r')
hold on
plot(t,y)
```

Primeiramente, é feita a leitura dos dados, depois foi implementado as equações do métodos das integrais. A partir disso foi obtido os seguintes resultados:

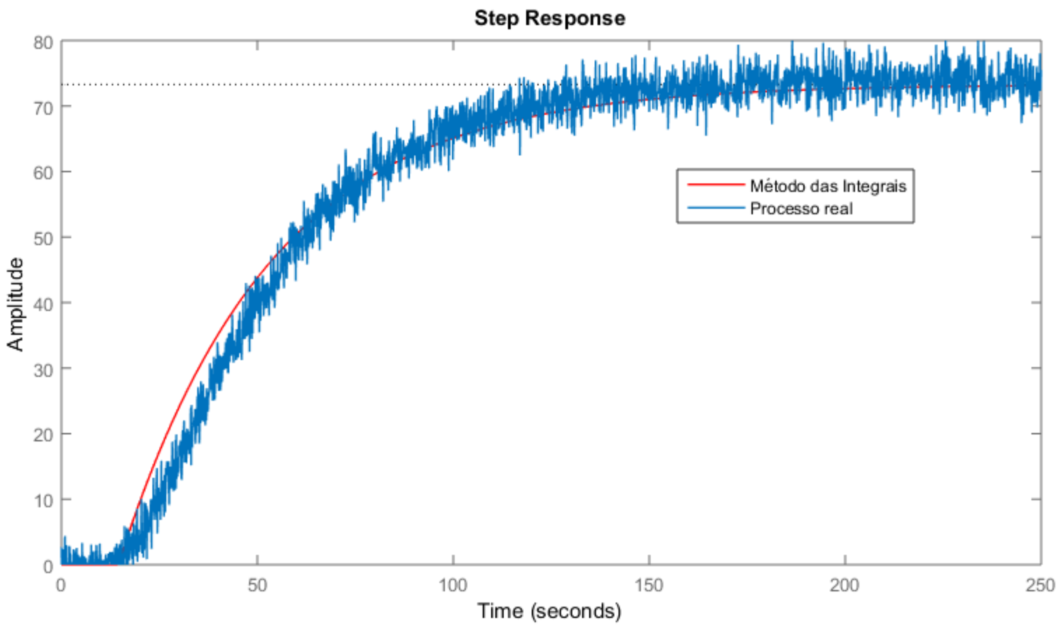

Figura 2: *Resposta para o método das Integrais.*

- Função de Transferência:

$$H1(s) = \frac{73.28 \cdot e^{-14.5s}}{38.97s + 1} \tag{2}$$

3.2 Método de uma constante de tempo (63,2 %)

Para aplicar o método de uma constante de tempo (63,2%) foi utilizado seguinte código no MATLAB:

```
% M todo da Constante de tempo (63,8%)
%%
dados_degrau = load('saidaLab.txt');
t = dados_degrau(:, 1);
u = dados_degrau(:,2);
y = dados_degrau(:,3);
%%
%Percebemos que a sa da tende ao seu valor permanente   partir do tempo
%220s, cujo  ndice    1100.

dy = mean(y(1100:3000)) - mean(y(1:86));
du = u(end)-0;
K = dy/du;
%% M todo de uma constante de tempo
%considerando o teta igual   22.4

yt1 = mean(y(1:86)) + (dy)*0.632;  %ponto que representa 63.2% do valor de Y

t1 = 0;
i =1;

%loop para encontrar o tempo 1:
while(y(i)<yt1)
    t1 = t(i);
    i = i+1;
end

%% Fun  es de trasfer ncia
s = tf('s');
teta = 18;
tal = t1-teta;

G1 = (K*exp(-teta*s))/(tal*s +1)
step(G1, 'r')
hold on
plot(t,y)
```

Novamente, é feita a leitura dos dados, depois foi implementado as equações do método de uma constante de tempo (63,2%). A partir disso foi obtido os seguintes resultados:

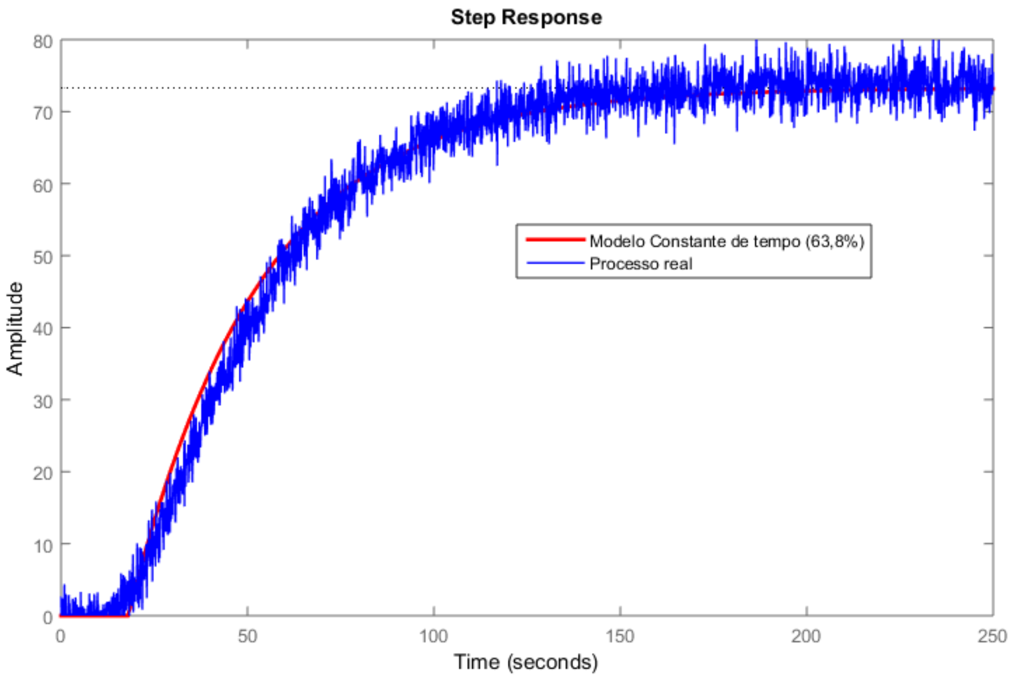

Figura 3: *Resposta para o método de uma Constante de tempo (63,2%).*

- Função de Transferência:

$$H2(s) = \frac{73.28 \cdot e^{-18s}}{35.4s + 1} \tag{3}$$

3.3 Método de quatro constantes de tempo (98 %)

Para aplicar o método de quatro constante de tempo (98%) foi utilizado o seguinte código no MATLAB:

```
% M todo Constante de Tempo (98%)
%%
dados_degrau = load('saidaLab.txt');
t = dados_degrau(:, 1);
u = dados_degrau(:,2);
y = dados_degrau(:,3);
%%
%Percebemos que a saida tende ao seu valor permanente a partir do tempo
%220s, cujo  ndice    1100.

dy = mean(y(1100:3000)) - mean(y(1:86));
du = u(end)-0;
K = dy/du;
%% M todo de 4 constante de tempo

yt1 =  mean(y(1:86)) + (dy)*0.983;  %Ponto que representa 63.2% do valor de Y

```

```
t1 = 0;
i =1;

%Loop para encontrar o tempo 1:

while(y(i)<yt1)
    t1 = t(i);
    i = i+1;
end
%% Fun  es de trasferencia

s = tf('s');
teta = 18;
tal = t1/4;
G1 = (K*exp(-teta*s))/(tal*s +1)
step(G1, 'r')
hold on
plot(t,y)
```

Novamente, é feita a leitura dos dados, depois foi implementado as equações do método de quatro constantes de tempo (98 %). A partir disso foi obtido os seguintes resultados:

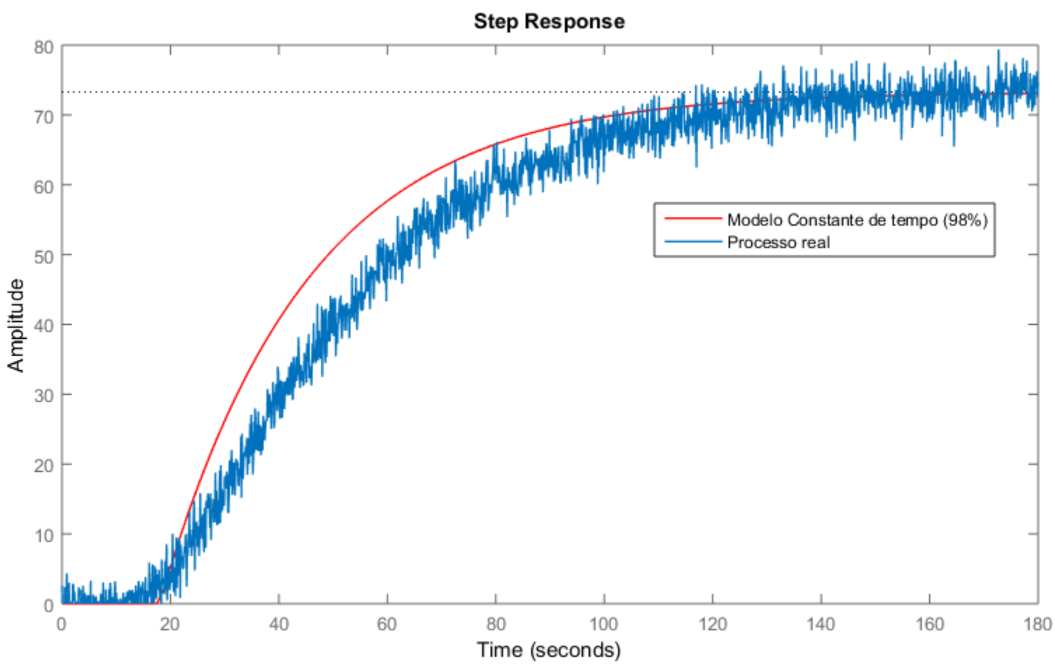

Figura 4: *Resposta para o método de quatro Constante de tempo (98%).*

- Função de Transferência:

$$H3(s) = \frac{73.28 \cdot e^{-18s}}{27.15s + 1} \tag{4}$$

3.4 Método de Ziegler-Nichols

Para aplicar o método de Ziegler- Nichols foi utilizado o seguinte código no MATLAB:

```
% m todo de Ziegler/Nichols%
%%
dados_degrau = load('saidaLab.txt');
t = dados_degrau(:, 1);
u = dados_degrau(:,2);
y = dados_degrau(:,3);
%%
%percebemos que a saida tende ao seu valor permanente a partir do tempo
%220s, cujo  ndice     1100.

dy = mean(y(1100:3000)) - mean(y(1:86));
du = u(end)-0;
K = dy/du;
%% Encontrando o ponto de inflexao
ypp = diff(y,2);

yppAbs = abs(ypp);
i = find(yppAbs == min(yppAbs))
t_infl = t(i); % tempo do ponto de inflexao;
y_infl =  y(i);  %valor de sa da do ponto de inflex o;
%% Criando a reta
t1 = 18;
i2 = 91;     %posi  o do tempo t1

%% Encontrando a reta

m = (y_infl - y(i2))/(t_infl - t(i2));

b= y_infl - ((y_infl - y(i2))/(t_infl -t(i2)))*t_infl;

max = dy + mean(y(1:86));


t3 = (max-b)/m;
%% Fun  es de trasferencia

s = tf('s');
teta = t1;
tal = t3-t1;
G1 = (K*exp(-teta*s))/(tal*s +1)
step(G1)
```

Novamente, é feita a leitura dos dados, depois foi implementado as equações do Método de

Ziegler-Nichols. A partir disso foi obtido os seguintes resultados:

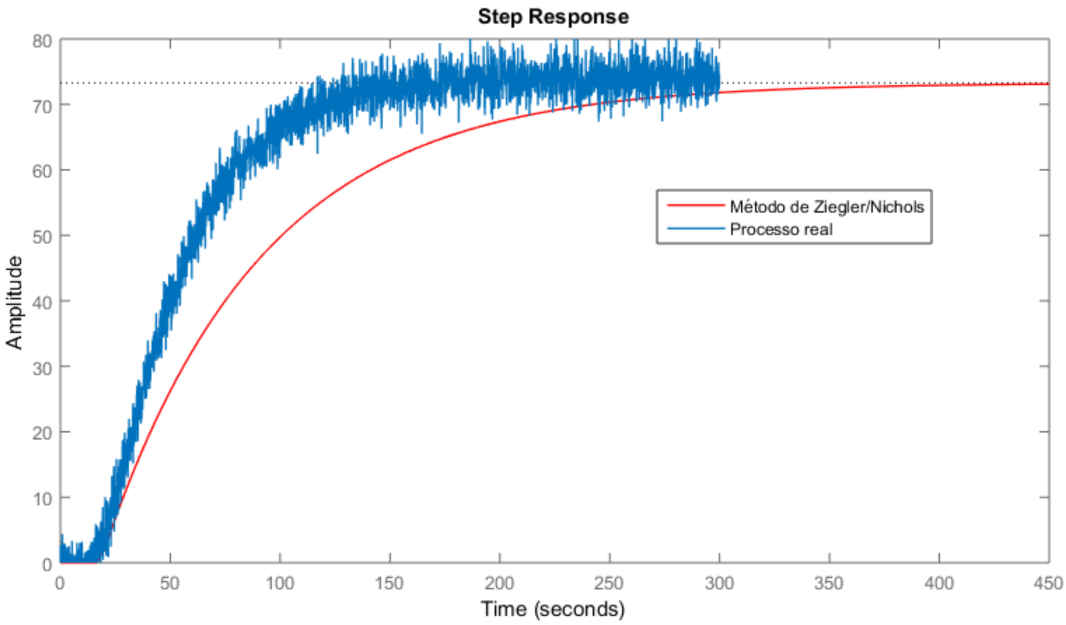

Figura 5: *Resposta para o método de Ziegler-Nichols.*

- Função de Transferência:

$$H4(s) = \frac{73.28 \cdot e^{-18s}}{72.2s + 1} \tag{5}$$

3.5 Método de Smith

Para aplicar o método de Smith foi utilizado o seguinte código no MATLAB:

```
% m todo de Smith
%%
dados_degrau = load('saidaLab.txt');
t = dados_degrau(:, 1);
u = dados_degrau(:,2);
y = dados_degrau(:,3);
%%
%percebemos que a saida tende ao seu valor permanente a partir do tempo
%220s, cujo indice e 1100.

dy = mean(y(1100:3000)) - mean(y(1:86));
du = u(end)-0;
K = dy/du;
%%
yt1 = mean(y(1:86)) + (dy)*0.283;  %ponto que representa 28.3% do valor de Y
yt2 = mean(y(1:86))+ (dy)*0.632;  %ponto que representa 63.2% do valor de Y

```

```
%%
t1 = 0;
t2 = 0;
i =1;

%loop para encontrar o tempo 1:
while(y(i)<yt1)
    t1 = t(i);
    i = i+1;
end

i =1;

%loop para encontrar o tempo 2:
while(y(i)<yt2)
    t2 = t(i);
    i = i+1;
end

%%
tal = 1.5*(t2-t1);
teta = t2 -tal;

s = tf('s');
G = (K*exp(-teta*s))/(tal*s +1)
step(G, 'r')
hold on
plot(t,y)
```

Novamente, é feita a leitura dos dados, depois foi implementado as equações do Método de Smith. A partir disso foi obtido os seguintes resultados:

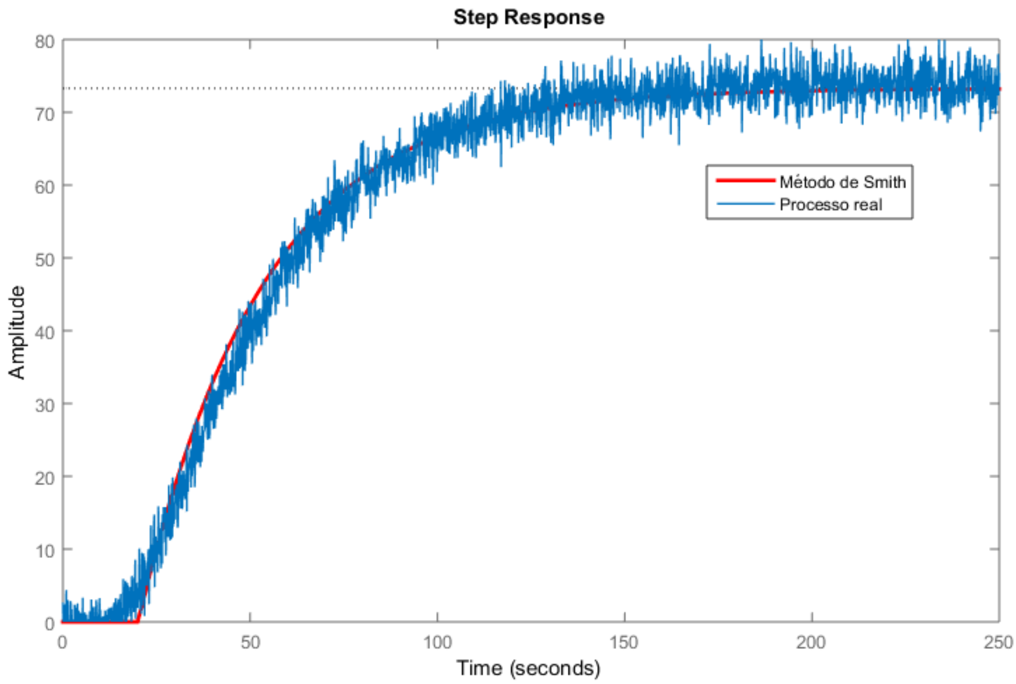

Figura 6: *Resposta para o método de Smith.*

- Função de Transferência:

$$H5(s) = \frac{73.28 \cdot e^{-20.1s}}{33.3s + 1} \tag{6}$$

3.6 Método de Haglund

Para aplicar o método de Haglund foi utilizado o seguinte código no MATLAB:

```
% m todo de haglund%
%%
dados_degrau = load('saidaLab.txt');
t = dados_degrau(:, 1);
u = dados_degrau(:,2);
y = dados_degrau(:,3);
%%
%percebemos que a saida tende ao seu valor permanente a partir do tempo
%220s, cujo indice e 1100.

dy = mean(y(1100:3000)) - mean(y(1:86));
du = u(end)-0;
K = dy/du;
%% Encontrando o ponto de inflex o
ypp = diff(y,2);

yppAbs = abs(ypp);
```

```
i = find(yppAbs == min(yppAbs))
t_infl = t(i); % tempo do ponto de inflex o;
y_infl =  y(i);  %valor de sa da do ponto de inflex o;
%% Criando a reta
teta = 18;
i_teta = 91;    %posi   o do tempo teta
m = (y_infl - y(i_teta))/(t_infl - t(i_teta));
b= y_infl - ((y_infl - y(i_teta))/(t_infl -t(i_teta)))*t_infl;


max = dy + mean(y(1:86));

t3 = (max-b)/m;
%% pegando o valor do tempo t2

t2 = (mean(y(1:86)) + (dy)*0.632 - b)/m;

%% Fun   es de trasferencia

s = tf('s');

tal = t2-teta;
G1 = (K*exp(-teta*s))/(tal*s +1)
step(G1, 'r')
hold on
plot(t,y)
```

Novamente, é feita a leitura dos dados, depois foi implementado as equações do método de Haglund. A partir disso foi obtido os seguintes resultados:

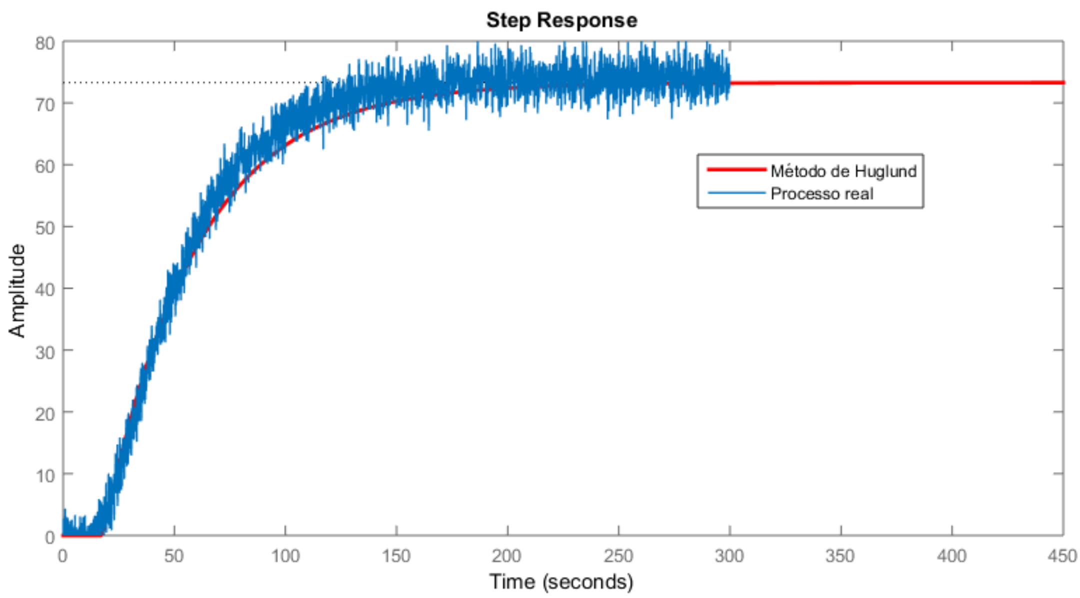

Figura 7: *Resposta para o método de Haglund.*

- Função de Transferência:

$$H6(s) = \frac{73.28 \cdot e^{-18s}}{41.46s + 1} \tag{7}$$

3.7 Método de Sundaresan e Krishnaswami

Para aplicar o método de Sundaresan e Krishnaswami foi utilizado seguinte código no MATLAB:

```
% m todo de Sundaresan
%%
dados_degrau = load('saidaLab.txt');
t = dados_degrau(:, 1);
u = dados_degrau(:,2);
y = dados_degrau(:,3);
%%
%percebemos que a saida tende ao seu valor permanente a partir do tempo
%220s, cujo  ndice    1100.

dy = mean(y(1100:3000)) - mean(y(1:86));
du = u(end)-0;
K = dy/du;
%%
yt1 = mean(y(1:86)) + (dy)*0.352;  %ponto que representa 35.3% do valor de Y
yt2 = mean(y(1:86)) + (dy)*0.853;  %ponto que representa 85.3% do valor de Y


%%
t1 = 0;
t2 = 0;
i =1;

%loop para encontrar o tempo 1:
while(y(i)<yt1)
    t1 = t(i);
    i = i+1;
end

i =1;

%loop para encontrar o tempo 2:
while(y(i)<yt2)
    t2 = t(i);
    i = i+1;
end
%%
tal = 0.67*(t2-t1);
```

```
teta = 1.3*t1 - 0.29*t2;

s = tf('s');
G = (K*exp(-teta*s))/(tal*s +1)
step(G, 'r')
hold on
plot(t,y)
```

Novamente, é feita a leitura dos dados, depois foi implementado as equações do métodos de Sundaresan. A partir disso foi obtido os seguintes resultados:

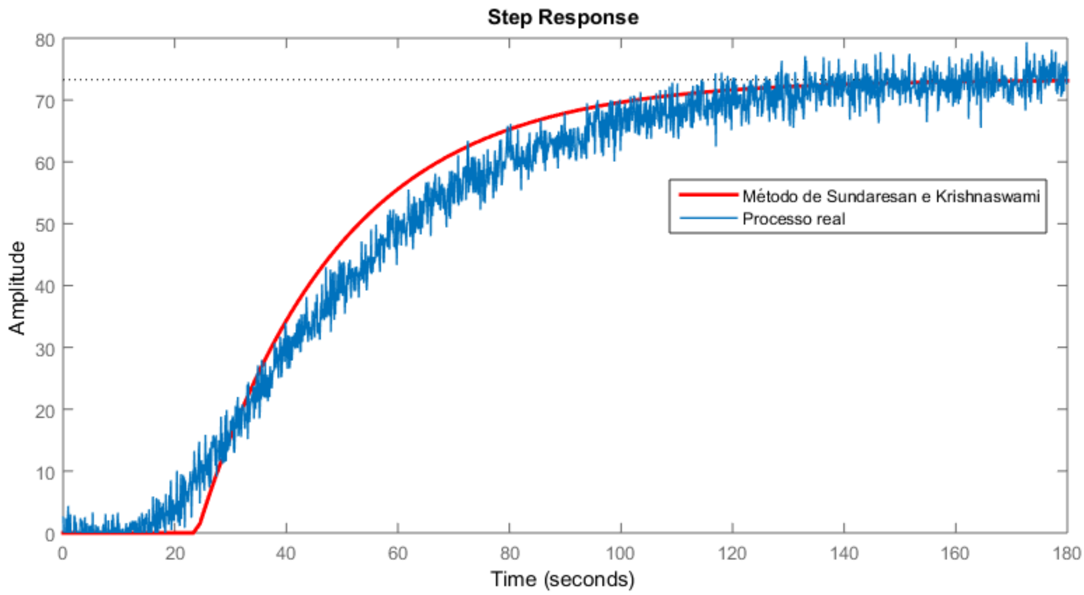

Figura 8: *Resposta para o método de Sundaresan e Krishnaswami.*

- Função de Transferência:

$$H7(s) = \frac{73.28 \cdot e^{-24s}}{25.33s + 1} \tag{8}$$

3.8 Método da Inclinação inicial

Para aplicar o método da inclinação inicial foi utilizado o seguinte código no MATLAB:

```
% m todo de Mollencamp
%%
dados_degrau = load('saidaLab.txt');
t = dados_degrau(:, 1);
u = dados_degrau(:,2);
y = dados_degrau(:,3);
%%
```

```
dy = mean(y(1100:3000)) -mean(y(1:86));
du = u(end)-0;

K = dy/du;
%%
yt1 = y(1) + (dy)*0.15;  %ponto que representa 15% do valor de Y
yt2 = y(1) + (dy)*0.45;  %ponto que representa 45% do valor de Y
yt3 = y(1) + (dy)*0.75;  %ponto que representa 75% do valor de Y


%%
t1 = 0;
t2 = 0;
t3 = 0;
i =1;

%loop para encontrar o tempo 1:
while(y(i)<yt1)
    t1 = t(i);
    i = i+1;
end

i =1;

%loop para encontrar o tempo 2:
while(y(i)<yt2)
    t2 = t(i);
    i = i+1;
end

i = 1

%loop para encontrar o tempo 2:
while(y(i)<yt3)
    t3 = t(i);
    i = i+1;
end

%%
x = (t2-t1)/(t3-t1);
lam = (0.0805-5.547*(0.475-x)^2)/(x-0.356);
f1 = 0.708*(2.811^lam);
w = f1/(t3-t1);
f2 = 0.922*(1.66^lam);
teta = t2-(f2/w);
tal1 = (lam+ sqrt(lam^2-1))/w
tal2 = (lam- sqrt(lam^2-1))/w

s = tf('s');
```

```
G = (K*exp(-teta*s))/((tal1*s +1)*(tal2*s+1))
step(G, 'r')
hold on
plot(t,y)
```

Novamente, é feita a leitura dos dados, depois foi implementado as equações do métodos da inclinação inicial. A partir disso foi obtido os seguintes resultados:

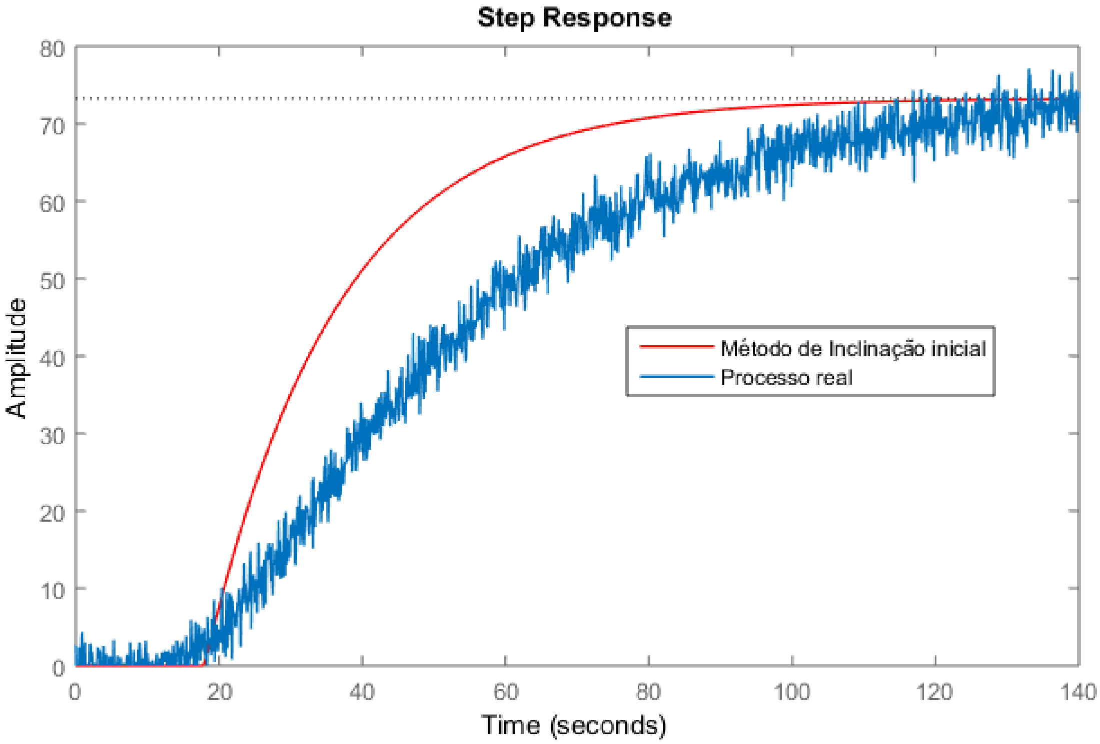

Figura 9: *Resposta para o método da Inclinação inicial.*

- Função de Transferência:

$$H8(s) = \frac{e^{-18s} \cdot 73.28}{18.41s + 1} \tag{9}$$

3.9 Comparação entre os Métodos

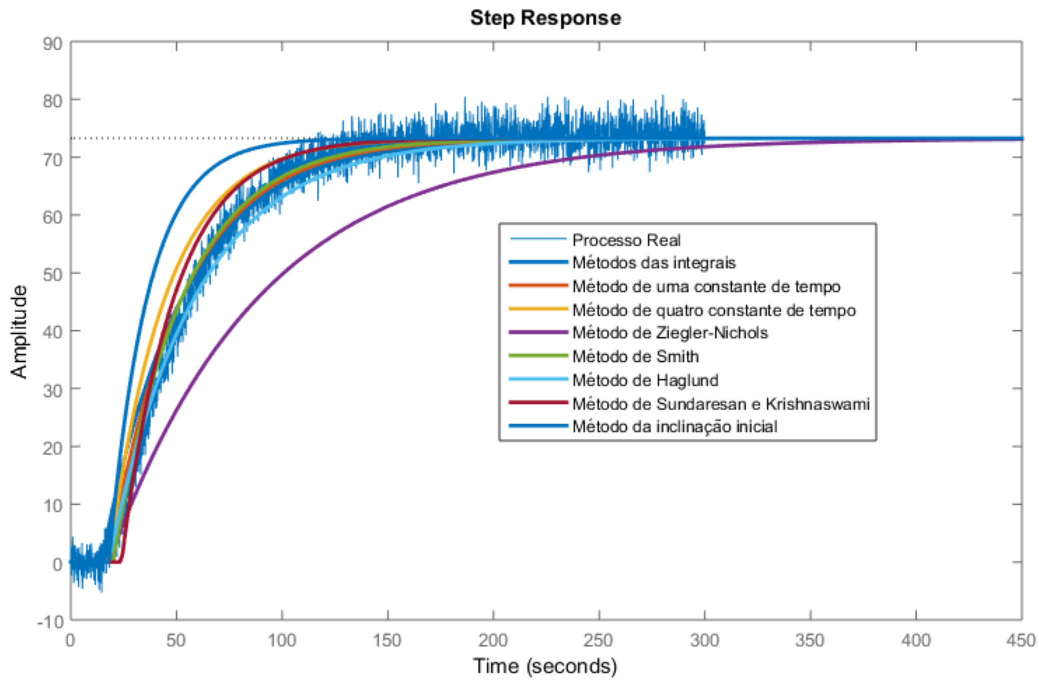

Figura 10: *Comparação entre os métodos de 1º ordem.*

Método	Tempo morto [s]	Constante de tempo [s]	Ganho [V/%]	Erro médio quadrático
Integrais	14.5	38.97	73.28	10.7015
Constante de tempo (63,2%)	18.0	35.40	73.28	7.7763
Constante de tempo (98%)	18.0	27.15	73.28	23.8738
Ziegler- Nichols	18.0	72.20	73.28	110.6986
Smith	20.1	33.30	73.28	7.0447
Haglund	18.0	41.46	73.28	8.8087
Sundaresan	24.0	25.73	73.28	14.0229
Inclinação Inicial	18.0	18.41	73.28	71.8362

Analisando as funções de transferência e os erros médios obtidos foi possível observar que os métodos de Ziegler-Nichols e da inclinação inicial foram que mais se distanciaram do esperado. Já os métodos de Smith e de uma constante de tempo foram os mais próximos. Nota-se que com um baixo valor na constante de tempo, mais rápido o modelo chegará em regime permanente, por outro lado, um alto valor demanda mais tempo para chegar em regime permanente.

4 Modelos de Segunda Ordem

A partir da equação 10 aplicável à sistemas de segunda ordem com atraso puro de tempo mostrada abaixo, podemos desenvolver os seguintes métodos:

- Método de Sundaresan;
- Método de Mollenkamp.

$$H(s) = \frac{k \cdot e^{-\Theta s}}{(\tau_1 s + 1)(\tau_2 s + 1)} = \frac{k w_n^2 \cdot e^{-\Theta s}}{s^2 + 2\zeta w_n + w_n^2} \tag{10}$$

4.1 Método de Sundaresan

Para aplicar o método de Sundaresan de segunda ordem foi utilizado o seguinte código no MATLAB, considerando a resposta à entrada ao degrau:

```
% Metodo de Sundaresan de Segunda Ordem
%%
dados_degrau = load('saidaLab.txt');
t = dados_degrau(:, 1);
u = dados_degrau(:,2);
y = dados_degrau(:,3);
%%
%Percebemos que a saida tende ao seu valor permanente a partir do tempo
%220s, cujo  ndice     1100.

dy = mean(y(1100:3000)) - mean(y(1:86));
du = u(end)-0;
K = dy/du;

%% normalizando o vetor
yN  =  y/K;
%%
teta_tal = trapz(t,u-yN);
indice=find(t <=teta_tal);
i  =585;

%%
tm=75; %tracei a reta tg e peguei o tempo do ponto que chega no maximo
mi=1/(tm-18);
l=(tm-teta_tal)*mi %usei para achar o n no grafico

n=0.6;
tal1= (n^(n/(1-n)))/mi;
tal2= (n^(1/(1-n)))/mi;
tald = teta_tal- tal1 - tal2;

```

```
%%
s = tf('s');
G = (K*exp(-tald*s))/((tal1*s +1)*(tal2*s+1))
step(G, 'r')
hold on
plot(t,y)
```

Novamente, é feita a leitura dos dados, depois foi implementado as equações do métodos de Sundaresan. A partir disso foi obtido os seguintes resultados:

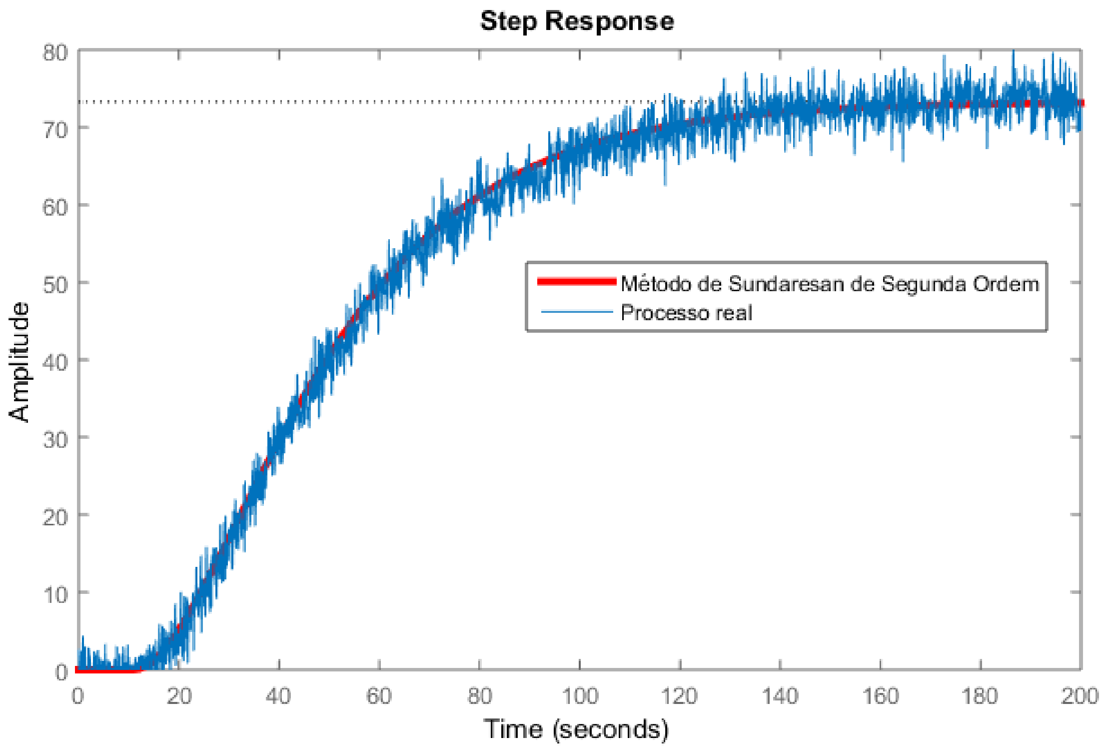

Figura 11: *Resposta para o método de Sundaresan.*

- Função de Transferência:

$$H9(s) = \frac{73.28 \cdot e^{-11.1s}}{421.1s^2 + 42.39s + 1} \tag{11}$$

4.2 Método de Mollenkamp

Para aplicar o método de Mollenkamp foi utilizado o seguinte código no MATLAB:

```
% metodo de Mollencamp
%%
dados_degrau = load('saidaLab.txt');
t = dados_degrau(:, 1);
```

```
u = dados_degrau(:,2);
y = dados_degrau(:,3);
%%
dy = mean(y(1100:3000)) -mean(y(1:86));
du = u(end)-0;

K = dy/du;
%%
yt1 = y(1) + (dy)*0.15;  %ponto que representa 15% do valor de Y
yt2 = y(1) + (dy)*0.45;  %ponto que representa 45% do valor de Y
yt3 = y(1) + (dy)*0.75;  %ponto que representa 75% do valor de Y


%%
t1 = 0;
t2 = 0;
t3 = 0;
i =1;

%loop para encontrar o tempo 1:
while(y(i)<yt1)
    t1 = t(i);
    i = i+1;
end

i =1;

%loop para encontrar o tempo 2:
while(y(i)<yt2)
    t2 = t(i);
    i = i+1;
end

i = 1

%loop para encontrar o tempo 2:
while(y(i)<yt3)
    t3 = t(i);
    i = i+1;
end

%%
x = (t2-t1)/(t3-t1);
lam = (0.0805-5.547*(0.475-x)^2)/(x-0.356);
f1 = 0.708*(2.811^lam);
w = f1/(t3-t1);
f2 = 0.922*(1.66^lam);
teta = t2-(f2/w);
tal1 = (lam+ sqrt(lam^2-1))/w
```

```
tal2 = (lam- sqrt(lam^2-1))/w

s = tf('s');
G = (K*exp(-teta*s))/((tal1*s +1)*(tal2*s+1))
step(G, 'r')
hold on
plot(t,y)
```

Novamente, é feita a leitura dos dados, depois foi implementado as equações do método de Mollenkamp. A partir disso foi obtido os seguintes resultados:

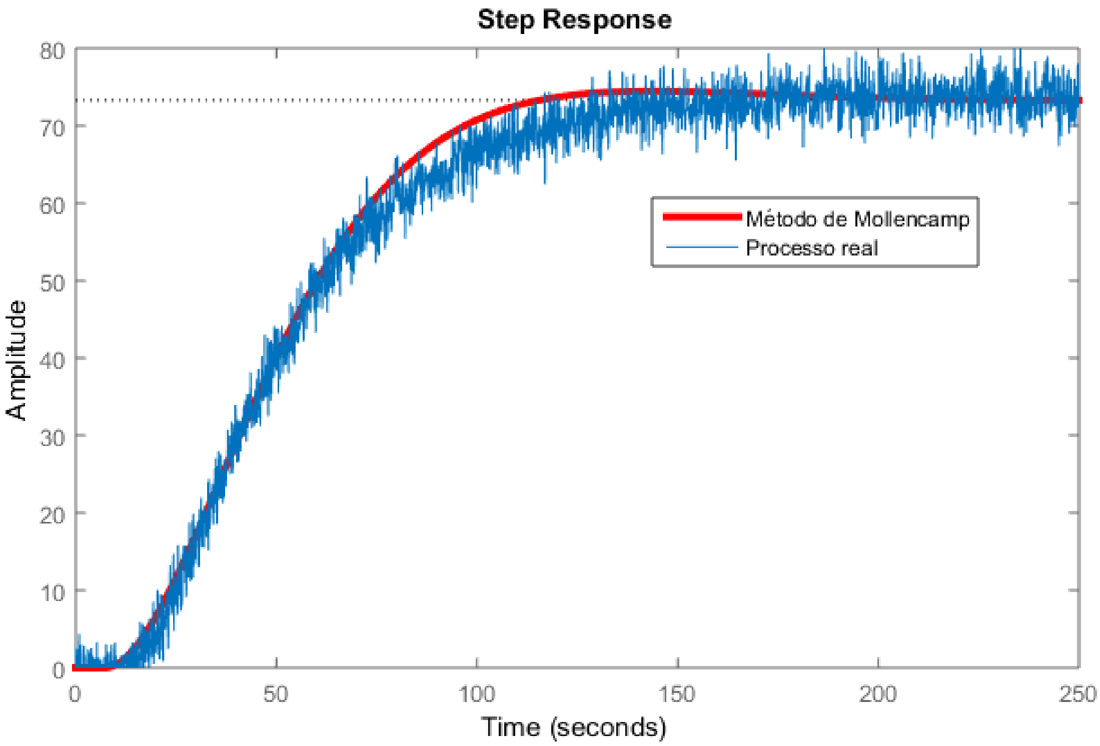

Figura 12: *Resposta para o método de Mollencamp.*

- Função de Transferência:

$$H10(s) = \frac{73.28 \cdot e^{-7.3s}}{687.2s^2 + 41.79s + 1} \tag{12}$$

4.3 Comparação entre os Métodos

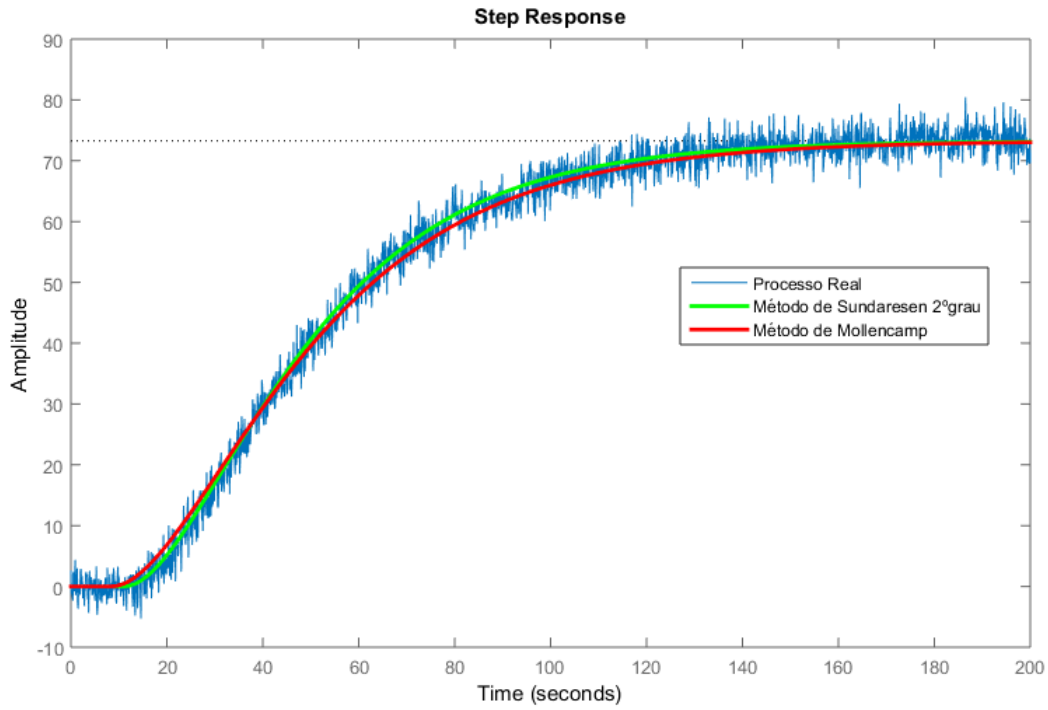

Figura 13: *Comparação entre os Métodos de 2º ordem.*

Método	**Tempo morto [s]**	**Constante de tempo 1 [s]**	**Constante de tempo 2 [s]**	**Ganho [V/%]**	**Erro médio quadrático**
Sundaresan	11.1	20.8943 + 15.8319 i	20.8943 - 15.8319 i	73.28	5.4174
Mollenkamp	07.3	26.4912	15.8947	73.28	5.6679

Analisando as funções de transferência e os erros médios obtidos foi possível observar que ambos métodos foram precisos e possuem uma ligeira diferença em seus valores.

5 Ensaio de entrada PRBS

A partir dos resultados obtidos no Labview para o ensaio de entrada PRBS, foi possível fazer a comparação entre os métodos anteriores utilizando o MATLAB, com o código a seguir:

```
% Ensaio com entrada PRBS
%%
s = tf('s')
G1 = exp(-14.5*s)*(73.28/(38.97*s + 1)); % Metodos das integrais
%%
G2 = exp(-18*s)*(73.28/(35.4*s + 1));     % Metodo de uma constante de tempo
%%
```

```
G3 = exp(-18*s)*(73.28/(27.15*s + 1));   % Metodo de quatro constante de tempo
%%
G4 = exp(-18*s)*(73.28/(72.2*s + 1));    % Metodo de Ziegler-Nichols
%%
G5 = exp(-20.1*s)*(73.28/(33.3*s + 1));  % Metodo de Smith
%%
G6 = exp(-18*s)*(73.28/(41.46*s + 1));   % Metodo de Haglund
%%
G7 = exp(-24*s)*(73.28/(25.33*s + 1));   % Metodo de Sundaresan e Krishnaswami;
%%
G8 = exp(-18*s)*(73.28/(18.46*s + 1)); % Metodo da inclina  o inicial
%%
G9 = exp(-11.1*s)*(73.28/(421.1*s^2 + 42.39*s + 1)); % Metodo de Sundaresen 2
    grau
%%
G10 = exp(-8.44*s)*(73.28/(507.4*s^2 + 46.55*s + 1)); % Metodo de Mollencamp
%%
figure(1)
step (G1,G2,G3,G4,G5,G6,G7,G8)

legend ('M todos das integrais','M todo de uma constante de tempo','M todo de
    quatro constante de tempo','M todo de Ziegler-Nichols','M todo de Smith','
    M todo de Haglund','M todo de Sundaresan e Krishnaswami','M todo da
    inclina  o inicial')
figure(2)
step(G9,G10)

legend('M todo de Sundaresen 2 grau ','M todo de Mollencamp')


%% Upload dos dados PRBS:
prbs = load('prbslab.txt');
t = prbs(:,1);
u = prbs(:,2);
y = prbs(:,3);

y1 = lsim(G1,u,t);    % Metodos das integrais
y2 = lsim(G2,u,t);    % Metodo de uma constante de tempo
y3 = lsim(G3,u,t);    % Metodo de quatro constante de tempo
y4 = lsim(G4,u,t);    % Metodo de Ziegler-Nichols
y5 = lsim(G5,u,t);    % Metodo de Smith
y6 = lsim(G6,u,t);    % Metodo de Haglund
y7 = lsim(G7,u,t);    % Metodo de Sundaresan e Krishnaswami;
y8 = lsim(G8,u,t);    % Metodo da inclinacao inicial
y9 = lsim(G9,u,t);    % Metodo de Sundaresen 2 grau
y10 = lsim(G10,u,t);  % Metodo de Mollencamp

%% Plotando as respostas:
figure(3)
```

```
plot(t, y, 'b');
hold on
plot(t,y1,'g');
plot(t,y2,'m');
plot(t,y3,'m');
plot(t,y4,'r');
plot(t,y5,'g');
plot(t,y6,'y');
plot(t,y7,'r');
plot(t,y8,'g');
plot(t,y9,'b');
plot(t,y10,'r');
title('Ensaio com a resposta PRBS')
xlabel('Tempo(s)')
ylabel('Tens o (V)')
legend ('Processo Real','M todos das integrais','M todo de uma constante de tempo','M todo de quatro constante de tempo','M todo de Ziegler-Nichols','M todo de Smith','M todo de Haglund','M todo de Sundaresan e Krishnaswami','M todo da inclina  o inicial','M todo de Sundaresen 2 grau ','M todo de Mollencamp')
grid on

%% Erro quadratico medio:

E1 = mean((y-y1).^2);  % Metodos das integrais
E2 = mean((y-y2).^2);  % Metodo de uma constante de tempo
E3 = mean((y-y3).^2);  % Metodo de quatro constante de tempo
E4 = mean((y-y4).^2);  % Metodo de Ziegler-Nichols
E5 = mean((y-y5).^2);  % Metodo de Smith
E6 = mean((y-y6).^2);  % Metodo de Haglund
E7 = mean((y-y7).^2);  % Metodo de Sundaresan e Krishnaswami;
E8 = mean((y-y8).^2);  % Metodo da inclinacao inicial
E9 = mean((y-y9).^2);  % Metodo de Sundaresen 2 grau
E10 = mean((y-y10).^2); % Metodo de Mollencamp

Erro = [E1 E2 E3 E4 E5 E6 E7 E8 E9 E10]
```

O resultado obtido está apresentado na figura abaixo:

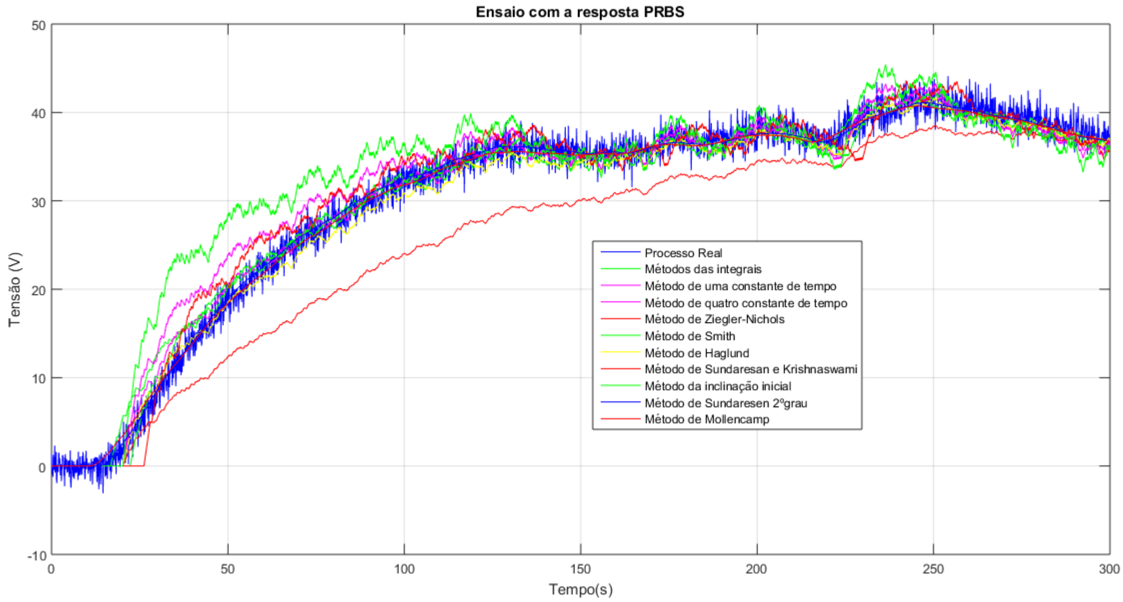

Figura 14: *Resposta para entrada PRBS.*

5.1 Primeira Ordem

Método	Tempo morto [s]	Constante de tempo [s]	Ganho [V/%]	Erro médio quadrático
Integrais	14.5	38.97	73.28	03.0706
Constante de tempo (63,2%)	18.0	35.40	73.28	02.4552
Constante de tempo (98%)	18.0	27.15	73.28	06.6253
Ziegler- Nichols	18.0	72.20	73.28	27.9464
Smith	20.1	33.30	73.28	02.4293
Haglund	18.0	41.46	73.28	02.6097
Sundaresan	24.0	25.73	73.28	04.7350
Inclinação Inicial	18.0	18.41	73.28	19.1076

5.2 Segunda Ordem

Método	Tempo morto [s]	Constante de tempo 1 [s]	Constante de tempo 2 [s]	Ganho [V/%]	Erro médio quadrático
Sundaresan	11.1	20.8943 + 15.8319 i	20.8943 - 15.8319 i	73.28	1.488
Mollenkamp	07.3	26.4912	15.8947	73.28	1.5962

Pode-se notar que os erros obtidos no ensaio PRBS para os sistemas de segunda ordem foram menores que os de primeira, uma vez que quanto maior a ordem, maior a sua complexidade e melhor será sua aproximação.

6 Considerações finais

A partir das análises feitas, é possível notar que os erros da entrada degrau foram maiores do que os do ensaio PRBS. Logo, este (PRBS) torna- se mais eficiente para sistemas ruidosos. Concomitantemente, ao observar a coluna dos erros dos métodos para segunda ordem do ensaio PRBS, concluímos que o método que apresentou menor valor foi o de Sundaresan. Podendo assim ser considerado o mais preciso.