EP28 - Projeto do controlador proporcional derivativo

Nome: Brayan Blail e Gabriel Pereira Silva

Seja o sistema dados pela FT de malha aberta $G(s) = \frac{2e^{-2s}}{5s+1}$

1) Projete um controlador PID via gráficos e Bode tal que UP=0 e ts<12s, detalhando os passos.

2) Mostre o gráfico de Bode original e compensado

3) Plote a resposta ao degrau atendendo a especificação.

4) Plote o sinal de controle.

O controlador PID é obtido projetando primeiro o controlador PI e depois o controlador PD, sendo

PI: $C(s) = K_p \cdot \frac{\left(s + \frac{K_i}{K_p}\right)}{s}$

PD: $C(s) = 1 + K_d \cdot s$

O controlador PI deve garantir o erro em regime e bons tempos de resposta. O controlador PD é usado para reduzir a sobreelevação, aumentando a margem de fase, que pode aparecer devido aos valores maiores de Kp para obter respostas rápidas.

PID: $C(s) = K_p\left(\frac{\left(s + \frac{K_i}{K_p}\right)}{s}\right)(1 + K_d \cdot s)$

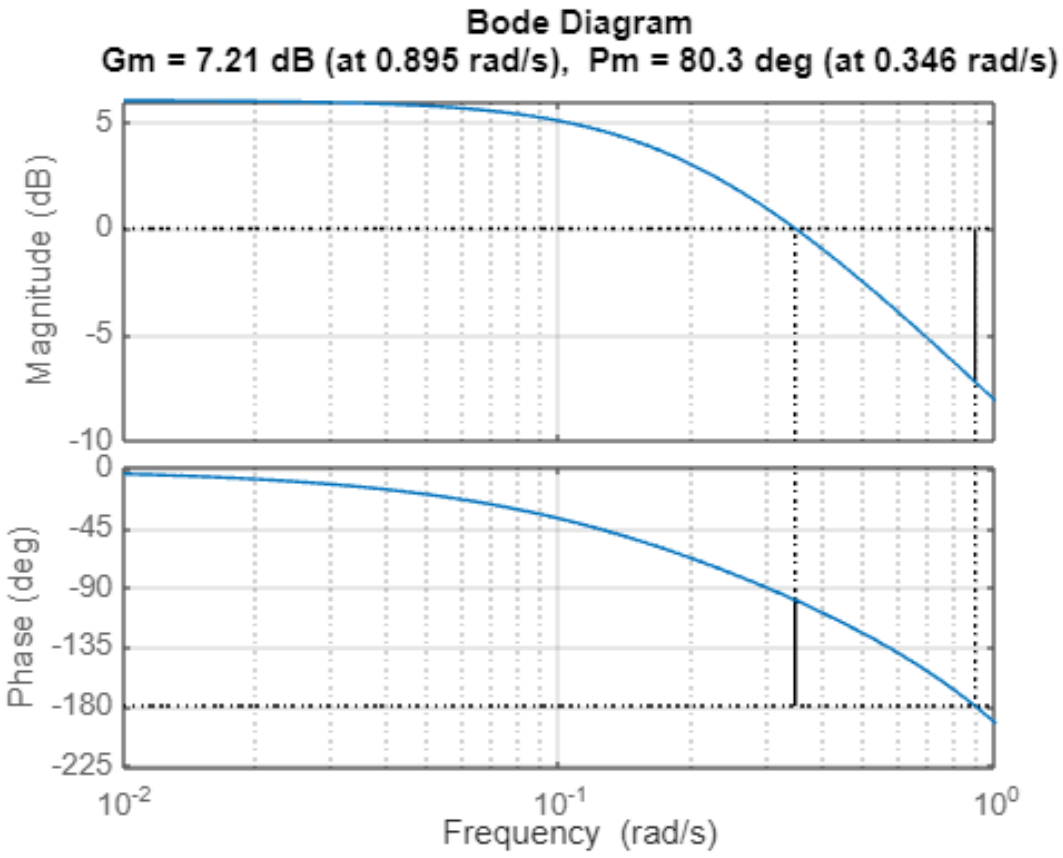
```
g = tf(2, [5 1],'InputDelay',2)
```

```
g =

                2
  exp(-2*s) * -------
              5 s + 1

Continuous-time transfer function.
Model Properties
```

```
w=logspace(-2,0,500);
margin(g,w);
grid;
```

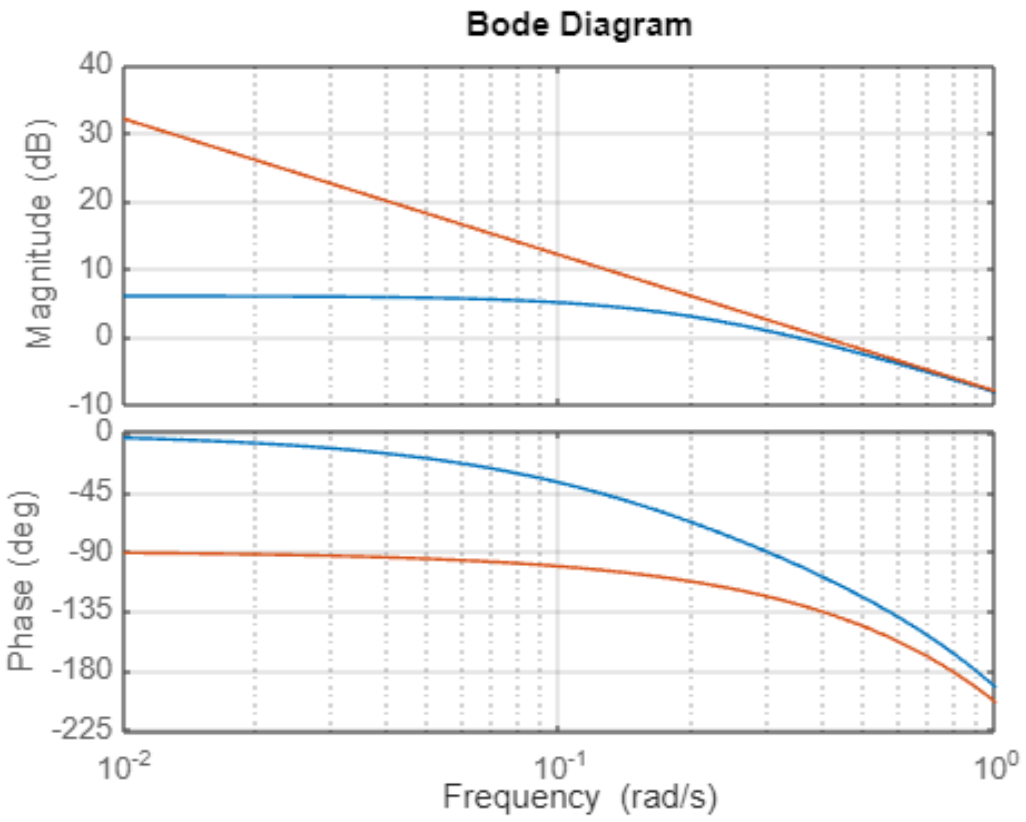


Vamos projetar um PI para atender tanto o tempo de estabelecimento ts<12s e erro em regime nulo**.** Para projetar um PI é necessário escolher um Kp incialmente e no nosso sistema a gente quer ter uma resposta rápida aumentando o Kp até um ponto de operação satisfatório. Inicialmente vamos escolher um Kp = 1 e colocar o zero do PI uma década antes do wg = 0.9, logo em wg´=0.1

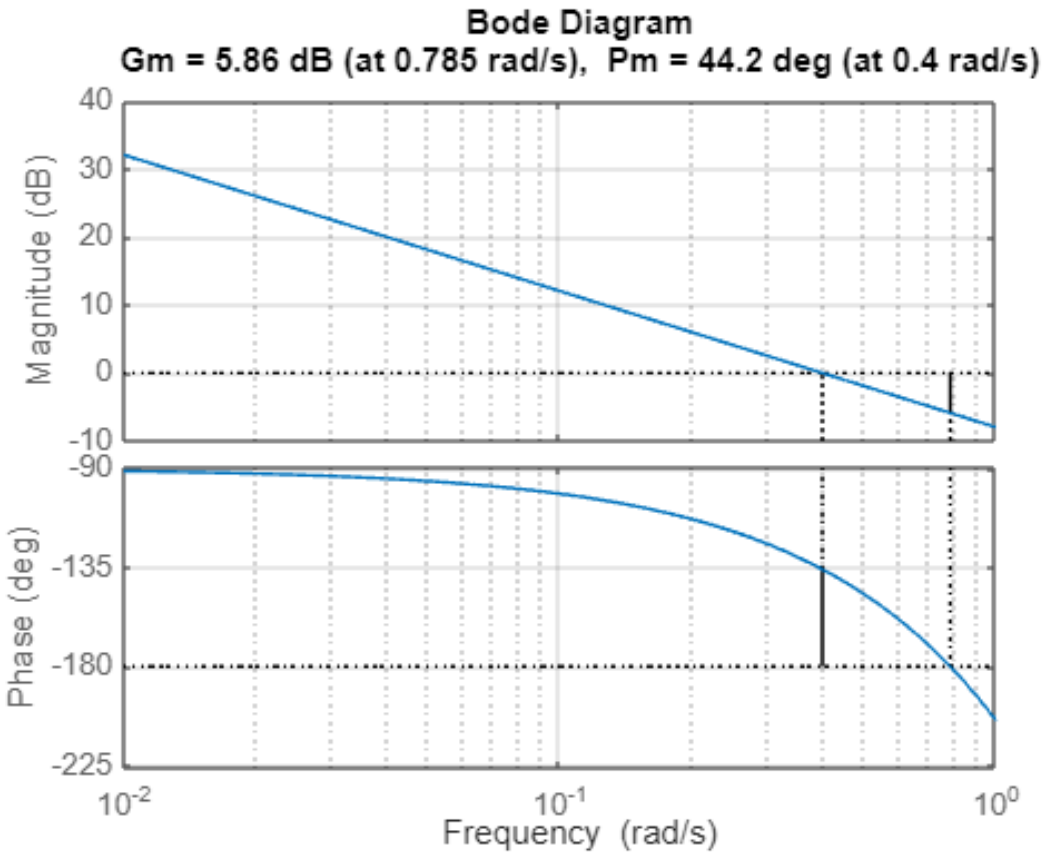
O zero do PI em $s = -\frac{K_i}{K_p}$ escolhemos colocar uma década antes de $\omega_g^{'}$, portanto em $0.1 rad/s$, de modo que em $\omega = 1 rad/s$ o efeito do atraso de fase do controlador já tenha desaparecido. Portanto,

$K_i = \frac{1}{10} K_p = 0.2$

```
Kp=1;
Ki=0.2;
c1=tf([Kp Ki],[1 0]);
bode(g,c1*g,w);grid;
```

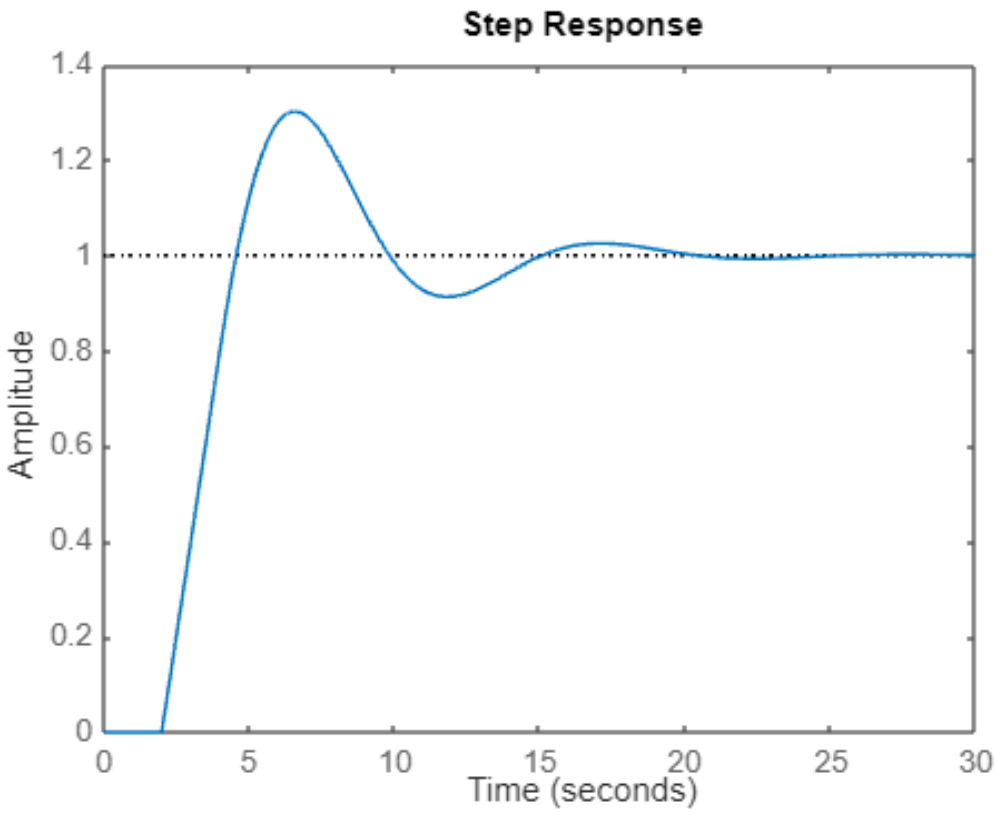



```
figure;
g2 = c1*g;
margin(g2,w); grid;
```



Na figura acima percebe-se que noso ganho como controlador aumentou 0.49 db e reduziu a margem de fase em 17.7º, como o nosso ganho é negativo quando a gente abaixa o Kp a gente está tendo um ganho maior e uma menor margem de fase e deixando o sistema mais rá

```
m_pi = feedback(c1*g,1);
step(m_pi)
```



Na figura acima apenas com o controlador PI a gente não atinge o tempo de estabelecimente e deixa a resposta bem oscilatória. Vamos prosseguir projetando um PD para resolver o problema da oscilação e analisar novamente a resposta.

```
kd=1.25;
c2 = tf([kd 1], 1)
```

```
c2 =

  1.25 s + 1

Continuous-time transfer function.
Model Properties
```

A partir da frequência do zero do PD, o módulo começa aumentar 20dB/década. Nesta frequência, a fase já avançou 45°.

Se o zero for colocado em $\omega_g^{'}$, o módulo não subirá nesta freqüência e a fase subirá 45°, aumentando a margem de fase. Logo, variações em torno desta freqüência permitirão obter a melhor MF possível. Então uma boa opção inicial é um zero próximo ao wg obtido no passo anterior. Escolhe-se wg=1

Vamos aumentar a margem de fase de sistema com o controlador PD colocando um zero torno de $\omega_g = 0.8 rad/s$, ou seja, $\frac{1}{K_d} = 0.8,$ Escolheremos $K_d = 1.25$. Logo, o controlador PD, é:

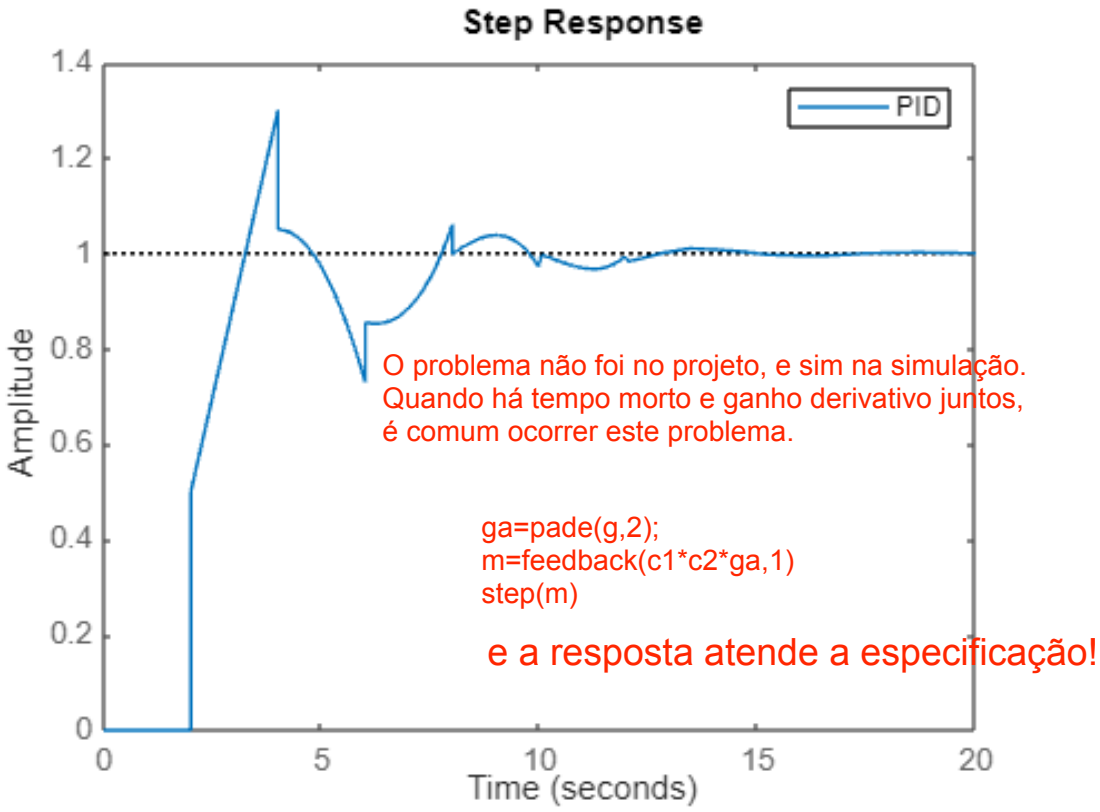
```
c3 = c1*c2
```

```
c3 =

  1.25 s^2 + 1.25 s + 0.2
  -----------------------
             s

Continuous-time transfer function.
Model Properties
```

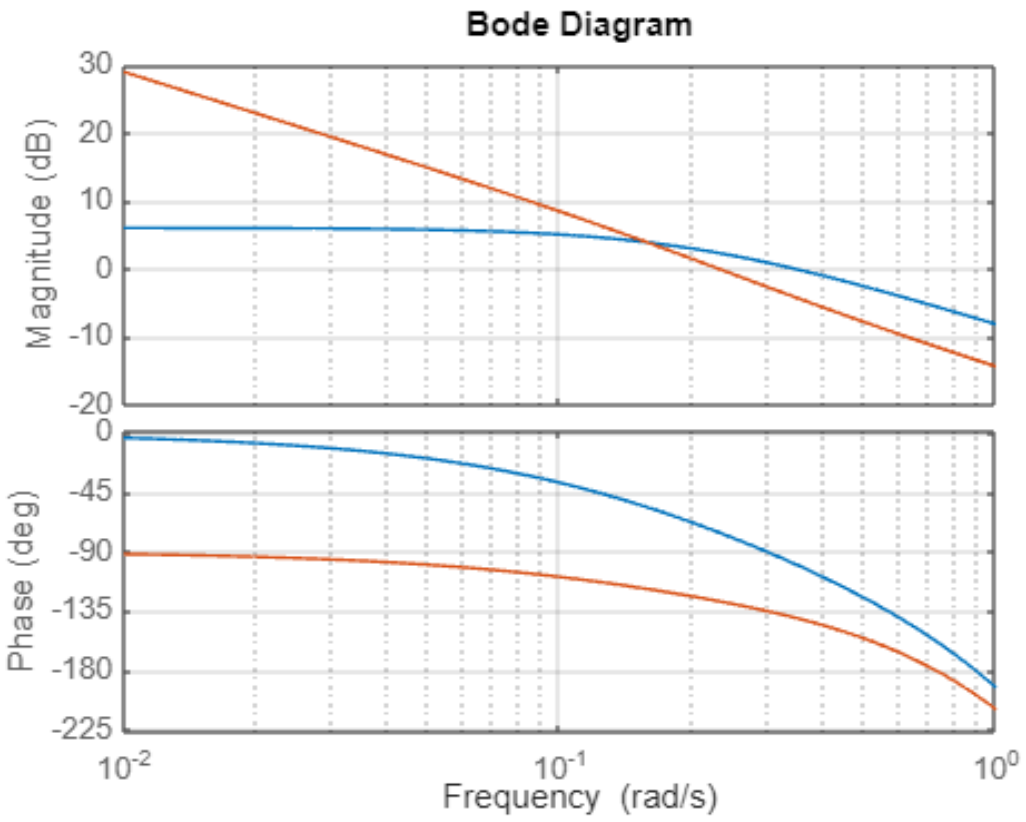
```
m2=feedback(c3*g,1);
figure;step(m2);legend('PID');
```



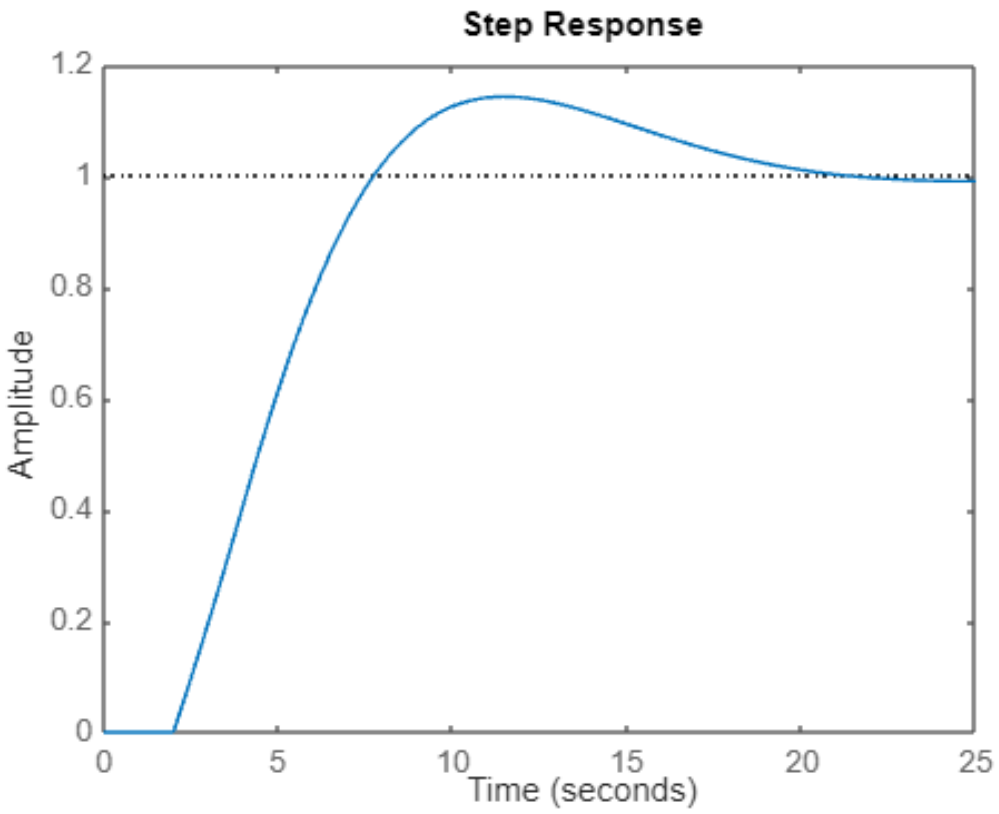
Com Kp=1 não atigimos o nosso objetivo mesmo variando o zero do PD em torno do wg.

Vamos fazer o PI com o Kp=0.47 e analisar a resposta novamente.

```
Kp2=0.47;
Ki2=0.14; % Zero do PI em 0.2979
c4=tf([Kp2 Ki2],[1 0]);
bode(g,c4*g,w);grid;
```




```
m_pi = feedback(c4*g,1);
step(m_pi)
```




```
L = stepinfo(m_pi);
L = L.SettlingTime
```

```
L = 19.1587
```

```
kd2=1.25;
c5 = tf([kd2 1], 1)
```

```
c5 =

  1.25 s + 1

Continuous-time transfer function.
Model Properties
```

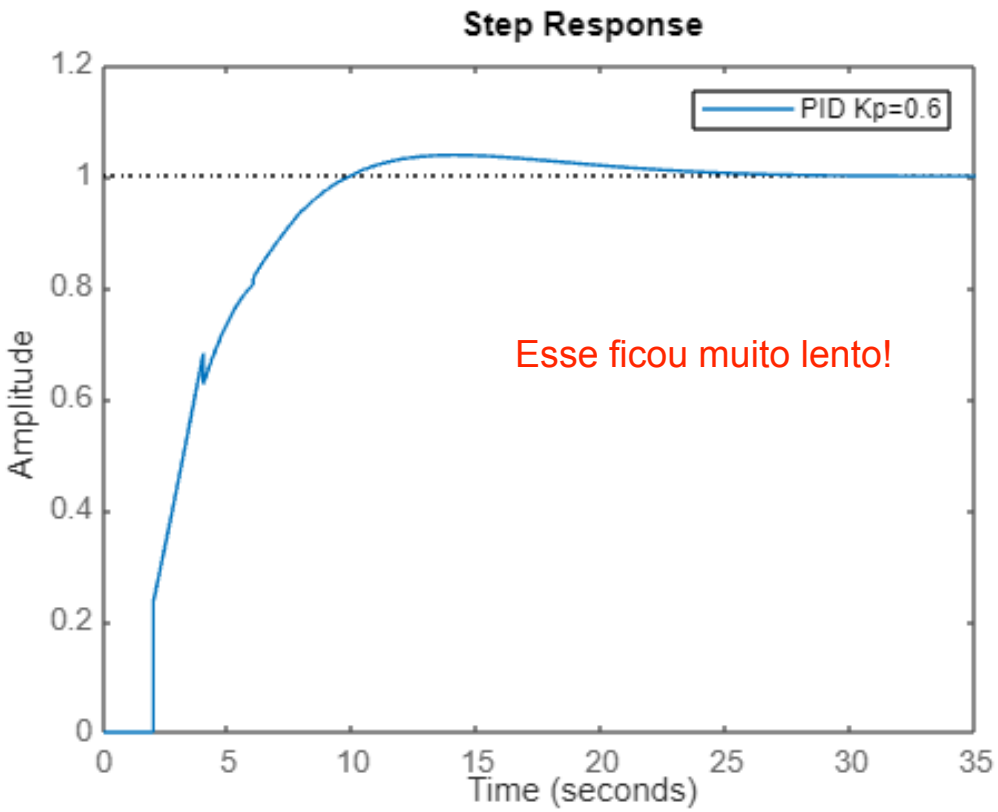
```
c6= c4*c5
```

```
c6 =

  0.5875 s^2 + 0.645 s + 0.14
  ---------------------------
              s

Continuous-time transfer function.
Model Properties
```

```
m3=feedback(c6*g,1);
figure;step(m3);legend('PID Kp=0.6');
```




```
T = stepinfo(m3);
T = T.SettlingTime
```

```
T = 19.6404
```

No final não conseguimos atingir o tempo de estabelecimento e overshoot par a função de tranferência dada, mesmo alterando diversos valor de Kp, Ki e Kd. Então projetamos o melhor controlador PID possível com a função de tranferência dada.