UFU 45 ANOS

Prova 02

Sistemas de Controle

Gabriel Resende Soares
11721ECP011

Izabela da Silva Neves
11811ECP026

Leonardo Vecchi Meirelles
12011ECP002

Lucas Humberto Jesus de Lima
12011ECP011

Novembro 2023

Sumário

Questão 1)

Considere o aeropêndulo apresentado na Figura 1 e Figura 2.

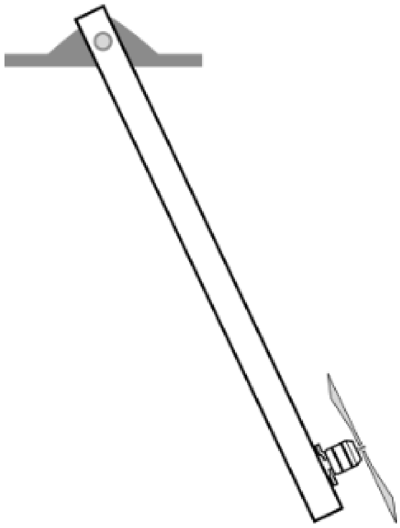


Figura 1 - Ilustração de um aeropêndulo.

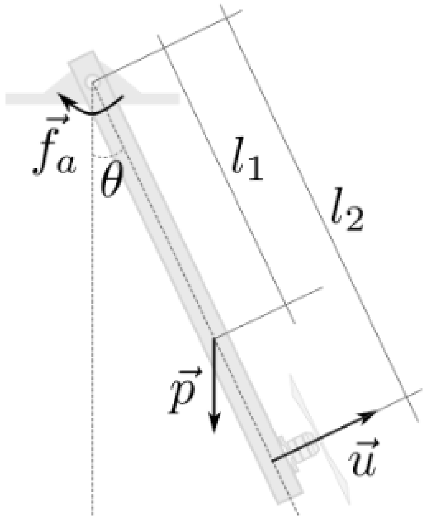


Figura 2 - Representação de forças e dimensões adotadas do aeropêndulo.

Com as seguintes dimensões:

$l_1 = 1{,}0\ m$
$l_2 = 1{,}35\ m$
$m = 1{,}25\ Kg$
$g = 9{,}81\ m/s^2$
$f_a = 1{,}5 \times 10^{-3}$
$J = 7{,}3 \times 10^{-3}\ Kg \cdot m^2$ - Momento de inércia do conjunto, com rotação

Letra a)

Encontre a equação não linear do sistema, determine a força de equilíbrio que equilibra o pêndulo em um ângulo θ_{eq} desejado e efetue a simulação em malha aberta com o sistema não linear começando em um ângulo $\theta_0 = 30°$.

Aplicando-se as leis de Newton ao diagrama de corpo livre, tem-se:

$$J\ddot{\theta} + f_a\dot{\theta} + pl_1 \sin(\theta) = ul_2$$

$$\therefore J\ddot{\theta} + f_a\dot{\theta} + mgl_1 \sin(\theta) = ul_2 \ \ (I)$$

Quando em equilíbrio, $\ddot{\theta} = \dot{\theta} = 0$, já que não há movimento. Assim, tem-se:

$$mgl_1 \sin(\theta) = ul_2$$

Atribuindo valores:

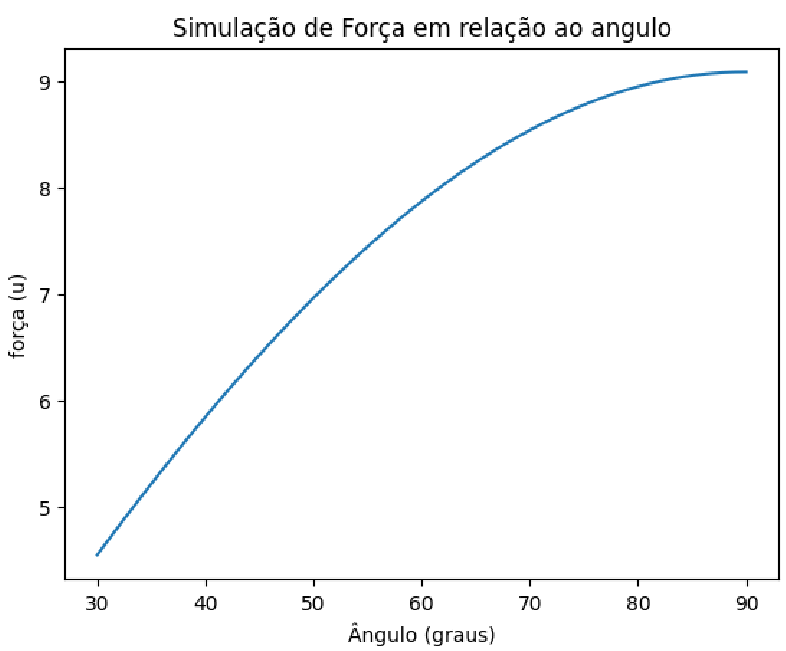
$$1{,}25 \cdot 9{,}81 \cdot 1 \cdot \sin(\theta) = u \cdot 1{,}35$$

$$9{,}083\sin(\theta) = u \ \ (II)$$

Com a fórmula obtida, fizemos um programa em Python para obter todas as forças(u) em variação do ângulo

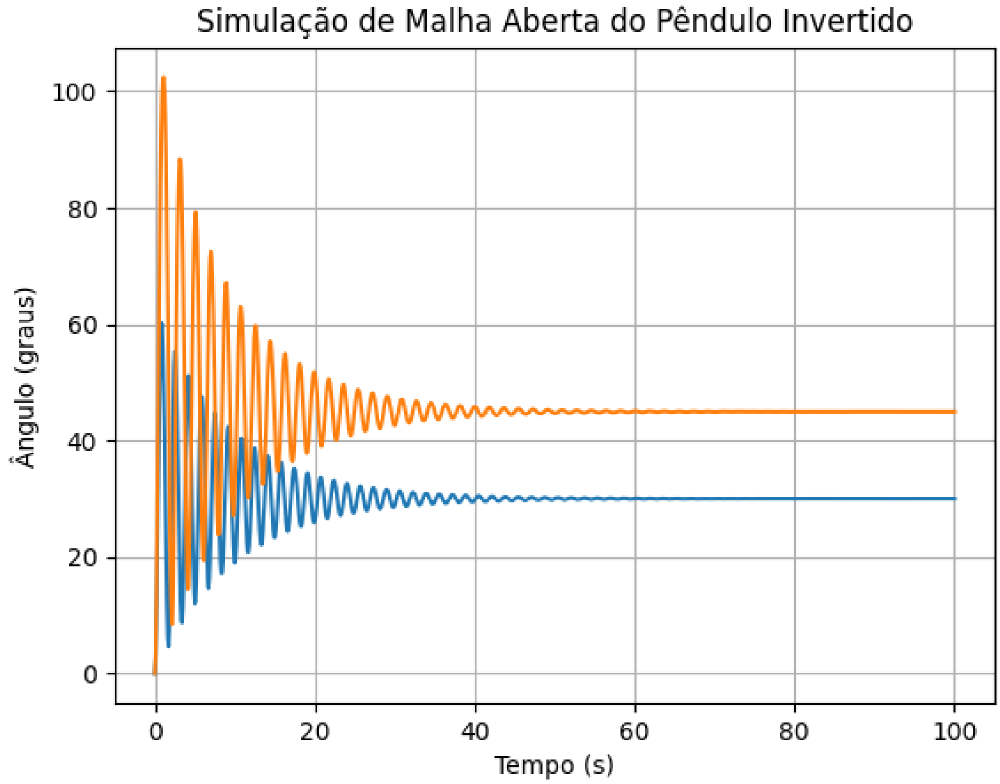
```
import math
import matplotlib.pyplot as plt

l = []
n = []
p = 30
while p <= 90:

  u = 9.083*math.sin((p*math.pi / 180.0 ))
  l.append(u)
  n.append(p)

  p += 1

plt.plot(n, l)
plt.ylabel('força (u)')
plt.title('Simulação de Força em relação ao angulo')
plt.xlabel('Ângulo (graus)')
plt.show()

print(l)
print(n)
```

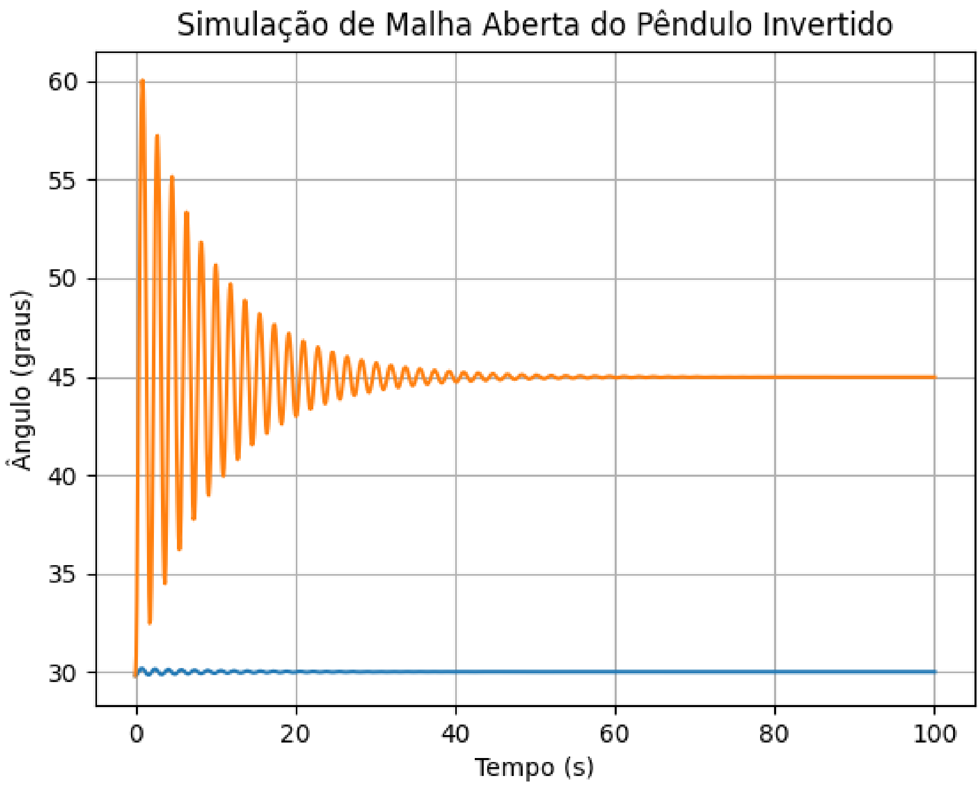
Gráfico obtido:



Com os valores obtidos podemos ver a relação do ângulo(θ) e a força(u) necessário para manter o sistema em equilíbrio, assim, com esses dados conseguimos montar um programa para simular o comportamento do ângulo(θ) em relação ao tempo(s)

```python
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

# Parâmetros do sistema
J = 0.0073
fa = 0.0015
l1 = 1
l2 = 1.35
m = 1.25
g = 9.81
u_30 = 4.54
u_45 = 6.42  # Aqui você pode ajustar o valor de u para malha aberta

# Condições iniciais
theta0 = 0.52  # Começando em trinta (em radianos)
theta_dot0 = 0.0

# Função que define o sistema de equações diferenciais
def pendulo_invertido(y, t, u):
    theta, theta_dot = y
    dydt = [theta_dot, (u*l2 - m*g*l1*np.sin(theta) - fa*theta_dot) / J]
    return dydt

# Tempo de simulação
t = np.linspace(0, 100, 1000)  # de 0 a 100 segundos, por exemplo

# Resolvendo as equações diferenciais
sol = odeint(pendulo_invertido, [theta0, theta_dot0], t, args=(u_30,))
sol1 = odeint(pendulo_invertido, [theta0, theta_dot0], t, args=(u_45,))


# Plotando os resultados
plt.plot(t, np.degrees(sol[:, 0]))
plt.plot(t, np.degrees(sol1[:, 0]))  # Convertendo de radianos para graus para o gráfico
plt.xlabel('Tempo (s)')
plt.ylabel('Ângulo (graus)')
plt.title('Simulação de Malha Aberta do Pêndulo Invertido')
plt.grid(True)
plt.show()
```


Simulação de Malha Aberta do Pêndulo Invertido
Ângulo (graus)
Tempo (s)
0
20
40
60
80
100


Simulação de Malha Aberta do Pêndulo Invertido
Ângulo (graus)
Tempo (s)
30
35
40
45
50
55
60
0
20
40
60
80
100

As simulações acima foram feitas a partir do código apresentado. As forças para os ângulos são obtidas a partir da equação (II). A fim de exemplificar, foram escolhidos valores de força para estabilização em 30° e 45°. Para valores altos de força, é possível que, pela inércia obtida, o pêndulo gire indefinidamente. Na primeira figura, o ângulo inicial é de 0 grau e na segunda é de 30 graus.

É perceptível que, com a força adequada, ao ser iniciado em 30 graus, o pêndulo se mantém estabilizado, como desejado.

Letra b)

Encontre a equação linearizada do sistema, obtenha a função de transferência e compare o comportamento temporal entre o sistema não linear e o sistema linearizado. Considera o movimento começando em um ângulo de $\theta_0 = 30°$.

Realizando a linearização da equação (I) de acordo com o método apresentado no livro do Nise, e considerando $\theta_0 = 30°$, encontra-se:

$$J\frac{d^2(\delta\theta + 30°)}{d\theta^2} + f_a\frac{d(\delta\theta + 30°)}{d\theta} + mgl_1 \sin(\delta\theta + 30°) = ul_2 \ (III)$$

Porém:

$$\frac{d^2(\delta\theta + 30°)}{d\theta^2} = \frac{d^2(\delta\theta)}{d\theta^2}$$

$$\frac{d(\delta\theta + 30°)}{d\theta} = \frac{d(\delta\theta)}{d\theta}$$

Expandindo $\sin(\delta\theta + 30°)$ pela série de Taylor truncada:

$$\sin(\delta\theta) = \left.\frac{dsin(\theta)}{d\theta}\right|_{\theta=30°} * \delta\theta = \cos(30°)\,\delta\theta = \delta\theta$$

$$\therefore J\frac{d^2(\delta\theta)}{d\theta^2} + f_a\frac{d(\delta\theta)}{d\theta} + mgl_1\,\delta\theta\frac{\sqrt{3}}{2} = ul_2 \ (IV)$$

A partir da equação linearizada, é possível obter a função de transferência Aplicando Laplace na equação IV:

$$J\,s^2(\Delta\Theta)_s + f_a s(\Delta\Theta)_s + mgl_1(\Delta\Theta)_s\sqrt{3}/2 = U_s l_2$$

$$(Js^2 + f_a s + mgl_1)(\Delta\Theta)_s = U_s l_2$$

$$\frac{(\Delta\Theta)_s}{U_s} = \frac{l_2}{Js^2 + f_a s + mgl_1 \sqrt{3}/_2}$$

Reorganizando os termos, tem-se:

$$\frac{(\Delta\Theta)_s}{U_s} = \frac{\frac{l_2}{J}}{s^2 + \frac{f_a}{J}s + \frac{mgl_1\sqrt{3}}{2J}}$$

Substituindo os valores dados e realizando as operações:

$$\frac{(\Delta\Theta)_s}{U_s} = \frac{\frac{1,35}{7,3 * 10^{-3}}}{s^2 + \frac{1,5 * 10^{-3}}{7,3 * 10^{-3}}s + \frac{12,2625\sqrt{3}}{2 * 7,3 * 10^{-3}}}$$

$$\therefore \frac{(\Delta\Theta)_s}{U_s} = \frac{184,9315}{s^2 + 0,2055s + 1454,7447}$$

Questão 2)

Para os vídeos apresentados na playlist:

https://www.youtube.com/playlist?list=PLxdnSsBqCrrF9KOQRB9ByfB0EUMwnLO9o

Assista aos vídeos:

20. Introduction to PID Control(opcional)

21. Practical Implementation Issues with a PID Controller(opcional)

22. Designing a PID Controller Using the Root Locus Method(obrigatório)

23. Designing a PID Controller Using the Ziegler-Nichols Method(obrigatório)

24. Using Root Locus to Meet Performance Requirements(obrigatório)

Utilizando um dos dois métodos apresentados (RootLocus ou Ziegler-Nichols), projete um controlador para o sistema apresentado na Questão 1. Utilize o ponto de equilíbrio de θ_{eq} = 30º.

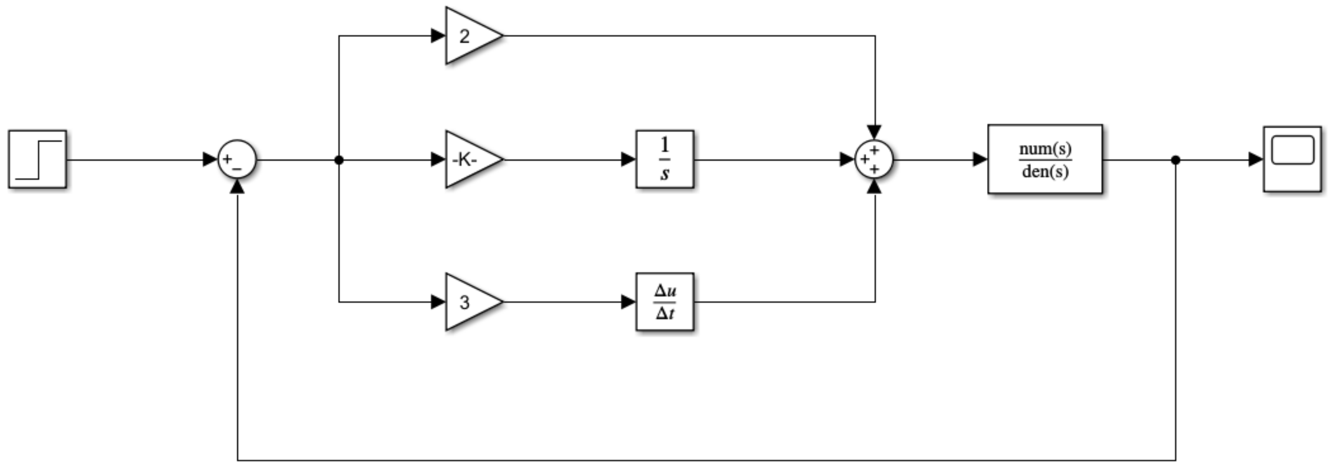
Letra a)

Utilizando um controlador PID e com ganhos do controlador limitado entre 0 e 100, determine o PID que resulte em, no máximo, 20% de ultrapassagem máxima e erro de regime permanente nulo.

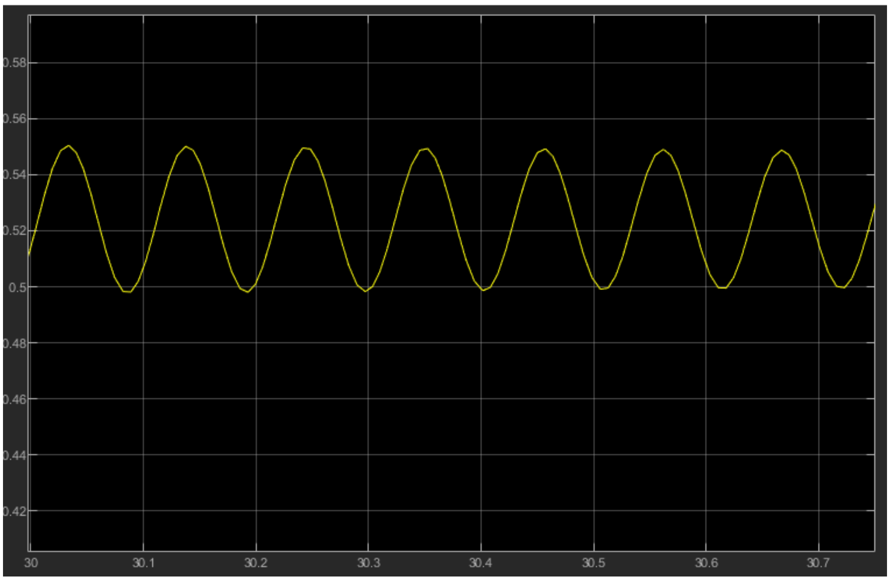
Pelo método de Ziegler-Nichols apresentado nos vídeos, foram calculados os parâmetros K_P, K_I e K_D de um controlador para o sistema da questão anterior.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt}e(t)$$

Primeiramente, o seguinte sistema foi montado no software Simulink:



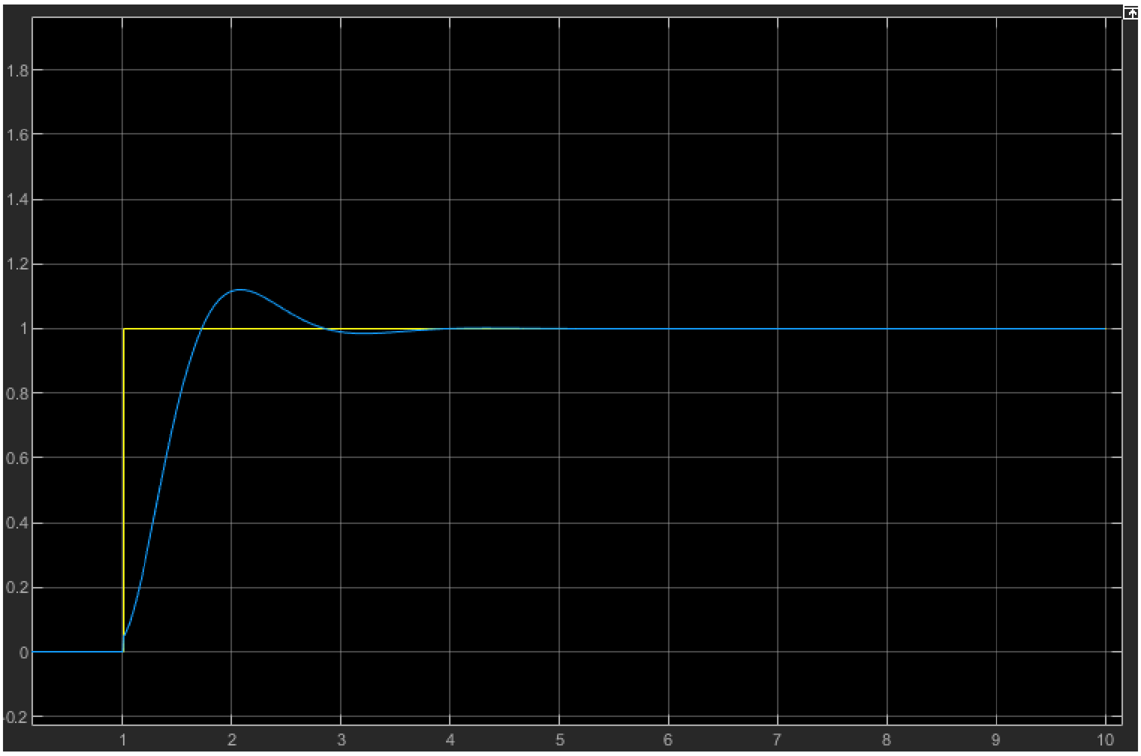
Em seguida, os valores de K_I e K_D foram zerados e estimou-se diferentes valores para K_P, avaliando o gráfico resultante, buscando uma estabilidade. Utilizando um ganho de 10, pode-se avaliar o seguinte comportamento:



A partir de tal gráfico, calculou-se seu período T_U, com o qual calculou-se os parâmetros K_P, K_I e K_D a partir da tabela abaixo:

Control Type	K_P	T_i	T_d	$K_I = K_P/T_i$	$K_D = T_d K_P$
PID (classic)	$0.6\ K_U$	$T_U/2$	$T_U/8$	$1.2K_U/T_U$	$0.075\ K_U T_U$
P	$0.5\ K_U$	-	-	-	-
PI	$0.45\ K_U$	$T_U/1.2$	-	$0.54K_U/T_U$	-
PD	$0.8\ K_U$	-	$T_U/8$	-	$0.1K_U T_U$
Pessen Integration	$0.7\ K_U$	$2T_U/5$	$3\ T_U/20$	$1.75K_U/T_U$	$0.105\ K_U T_U$
Some Overshoot	$K_U/3$	$T_U/2$	$T_U/3$	$(2/3)K_U/T_U$	$(1/9)K_U/T_U$
No Overshoot	$0.2\ K_U$	$T_U/2$	$T_U/3$	$(2/5)K_U/T_U$	$(1/15)K_U/T_U$

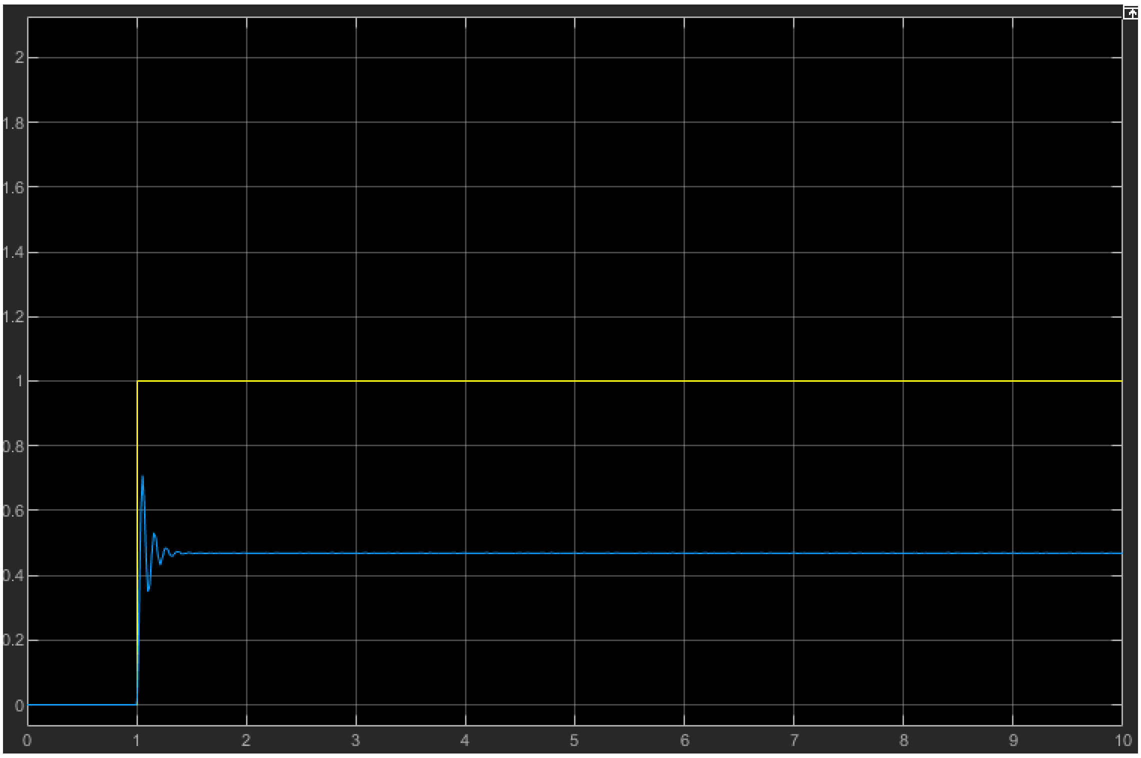
Com os valores encontrados de $K_P = 2$, $K_I = 33,3334$ e $K_D \approx 3$ obteve-se o seguinte gráfico, seguindo os requisitos passados na questão:



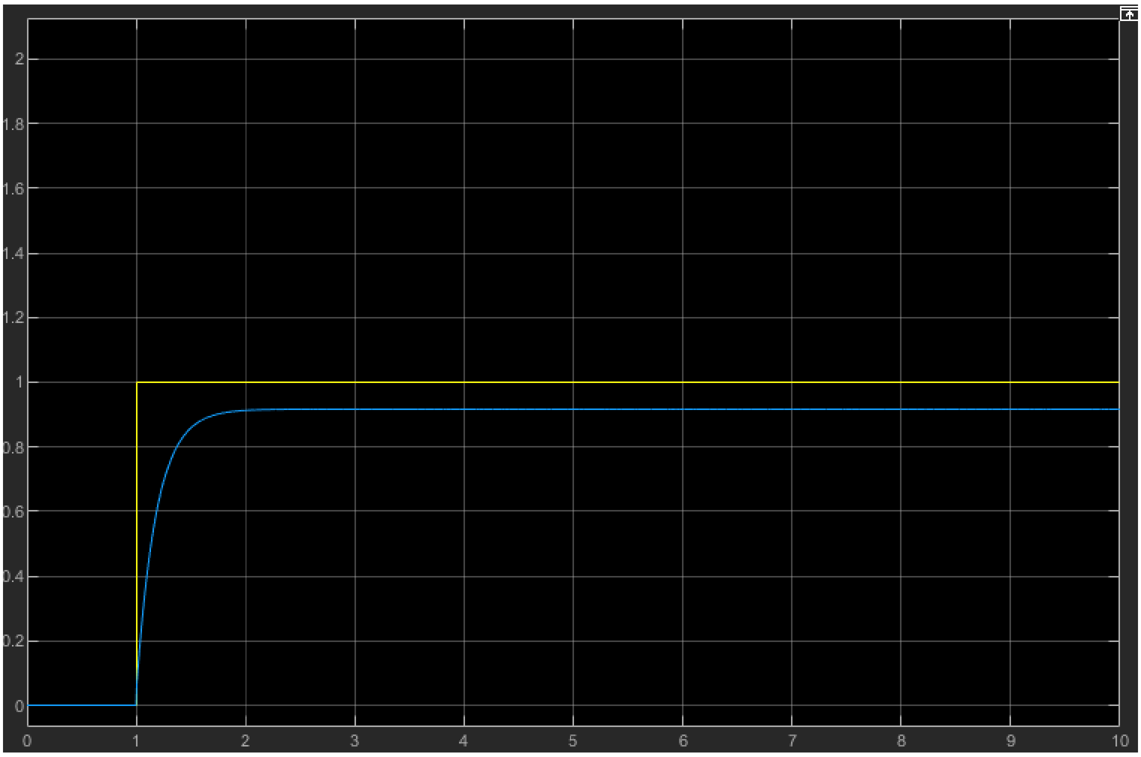
Letra b)

Verifique a possibilidade de projeto de um controlador PD (Proporcional-Derivativo), com ganhos do controlador limitados entre 0 e 100, que resulte em um sistema com, no máximo, 25% de ultrapassagem máxima e erro de regime permanente nulo. Justifique sua resposta.

Utilizando a tabela apresentada anteriormente, foram calculados os valores para K_P e K_D resultando no seguinte gráfico:



Como o erro de regime permanente não foi nulo, foram testados maiores ganhos para K_P e K_D, e, como esperado, a alteração em K_D causou a diminuição do overshooting e a alteração de K_P trouxe o valor de regime permanente mais próximo ao desejado, porém ainda com erro, como apresentado no gráfico abaixo:



Dessa forma, foi demonstrado que, como esperado, um controlador PD por si só não pode eliminar completamente o erro de regime permanente. Isso acontece pois o componente integrador do sistema é responsável por eliminar tal erro.