Lista 2

Luiz Georg 15/0041390

October 27, 2021

Setup

[1]:
```
import numpy as np
import control

np.set_printoptions(precision=4)
```

Lista 2

Condições do problema

[2]:
```
# Modelo da Planta
# x_ = A @ x + B @ d

A = np.array([[-4.1172, 0.7781, 0.0], [-33.8836, -3.5729, 0.0], [0.0, 1.0, 0.0]])

B = np.array([[0.5435, -39.0847, 0.0]]).T

x0 = np.array([[0.5, 0, -0.1]]).T

print("A:")
print(A)
print()

print("B:")
print(B)
print()

print("x0:")
print(x0)
```

```
A:
[[ -4.1172   0.7781   0.    ]
 [-33.8836  -3.5729   0.    ]
 [  0.       1.       0.    ]]

B:
[[  0.5435]
 [-39.0847]
 [  0.    ]]

x0:
[[ 0.5]
 [ 0. ]
 [-0.1]]
```

1 Quais são os autovalores de A? O sistema é estável, instável ou marginalmente estável? Subamortecido ou superamortecido?

[3]:
```
# Autovalores de A
eigvals = np.linalg.eigvals(A)
print("polos:")
print(np.reshape(eigvals, (3, 1)))
```

```
polos:
[[ 0.    +0.j    ]
 [-3.8451+5.1275j]
 [-3.8451-5.1275j]]
```

Sistema marginalmente estável (2 polos no semiplano negativo e um polo na origem);
Sistema subamortecido (par de polos complexos).

2 O sistema é controlável ou não? Justifique.

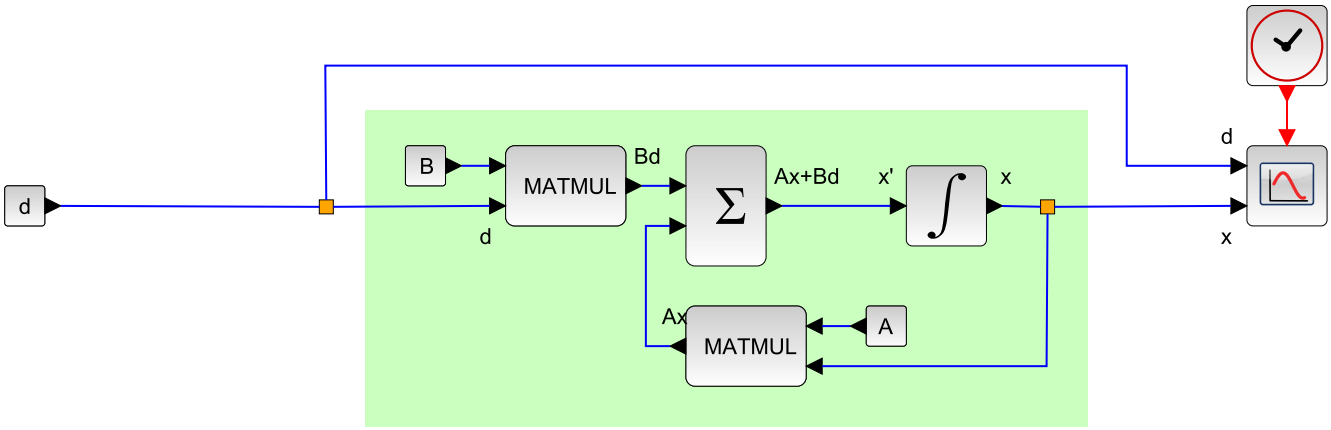
[4]:
```
# Matriz de controlabilidade
R = np.hstack([B, A @ B, A @ A @ B])
print(np.linalg.matrix_rank(R) == A.shape[1])
```

```
True
```

Sistema completamente controlável, pois o rank da matriz de controlabilidade é igual ao número de estados.

3 Modele o sistema dinâmico no Simulink (MATLAB), Xcos (Scilab), ou similares

Diagrama do sistema (em verde) e sistema de simulação implementado no software xcos (scilab). Em todos os modelos desse documento, os valores do projeto foram calculados utilizando o código desse documento e copiados para o modelo no software xcos conforme necessário.

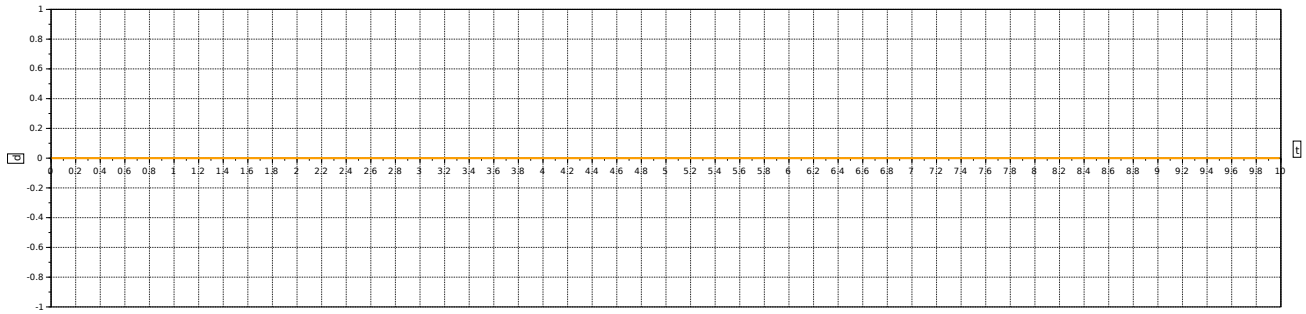


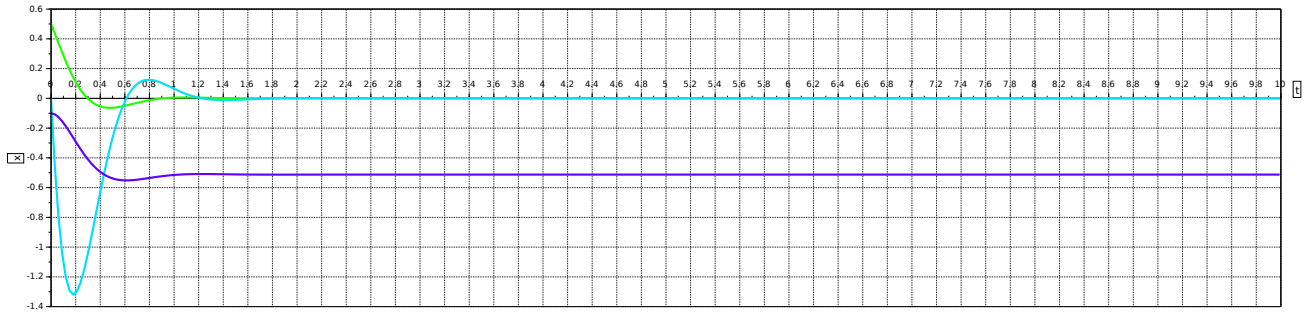
4 Simule o sistema nos seguintes casos

4.1 Entrada Nula

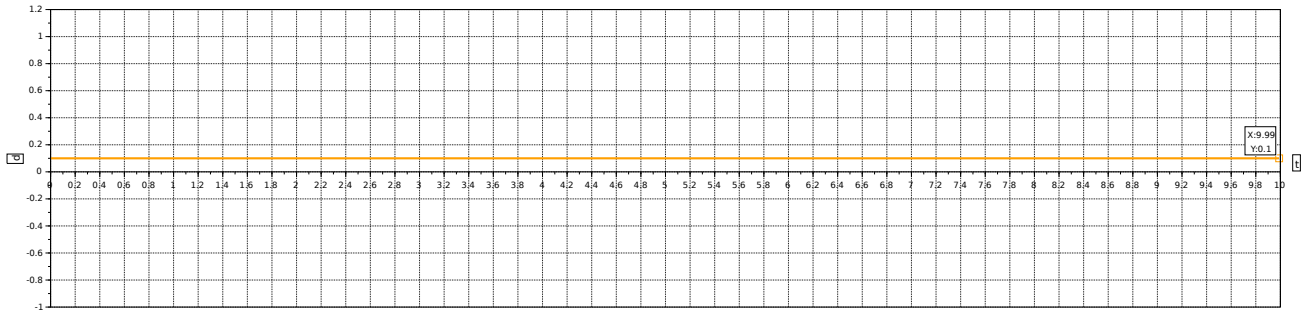
A imagem abaixo apresenta a simulação da resposta no tempo do sistema para entrada nula.

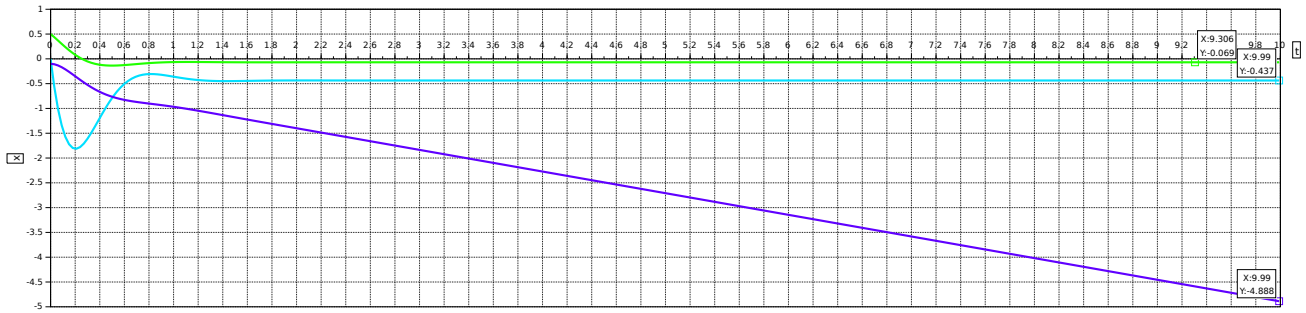
Nesse e em todos os próximos gráficos de resposta no tempo nesse documento, a linha laranja no gráfico superior representa a entrada na planta δ (chamada de d no gráfico por limitações do software), enquanto as linhas verde, azul e ciano, respectivamente, representam os estados α, q, e θ no gráfico de baixo. O eixo horizontal em todos os gráficos vai de 0 a 10 segundos, enquanto o eixo vertical foi ajustado automaticamente para destacar os detalhes do gráfico. Comparações de magnitude dever ser feitas com observação cuidadosa da escala de cada gráfico





4.2 Entrada degrau de 0.1 rad





5 Projete um regulador via alocação de polos (ou seja, calcule o vetor K que forneça um sistema com as seguintes características de malha fechada

- Sobressinal de 20%
- Tempo de pico de 1 segundo

[5]:
```
# Escolha de polos
tp = 1
PO = 0.20

wd = np.pi * tp
sigma = -abs(wd * np.log(0.2) / np.pi)
```

```
print(f"wd: {wd}")
print(f"sigma: {sigma}")
```

```
wd: 3.141592653589793
sigma: -1.6094379124341
```

[6]:
```
# Ganho do feedback completo
K = control.place(A, B, (sigma + wd * 1j, sigma - wd * 1j, 10 * sigma))
print("K:")
print(K)
```

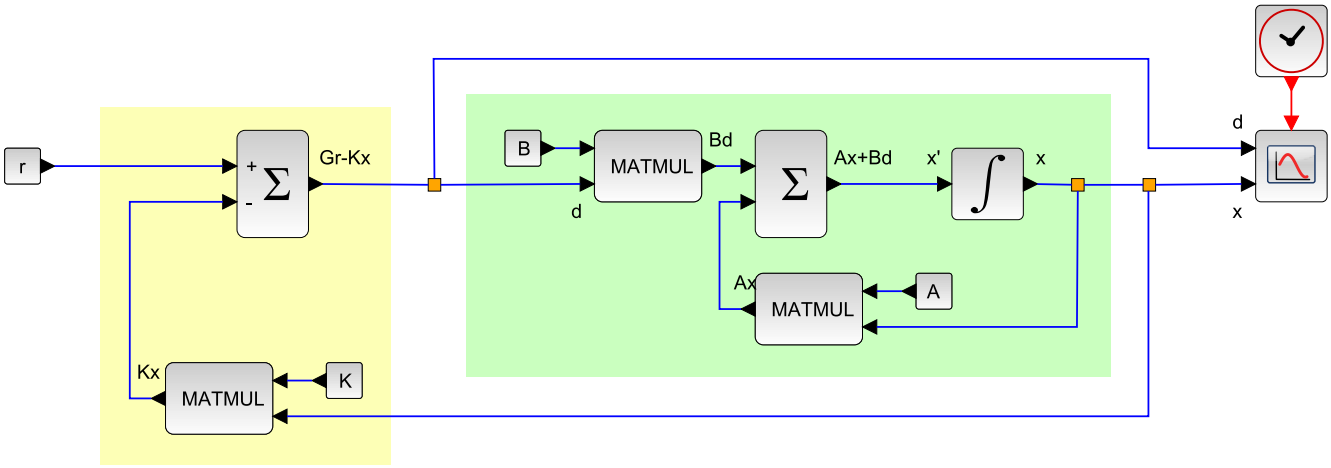
```
K:
[[ 2.3849 -0.2642 -1.1182]]
```

[7]:
```
# Autovalores resultantes (conferir alocação)
eigvals = np.linalg.eigvals(A - B @ K)
print("polos:")
print(np.reshape(eigvals, (3, 1)))
```

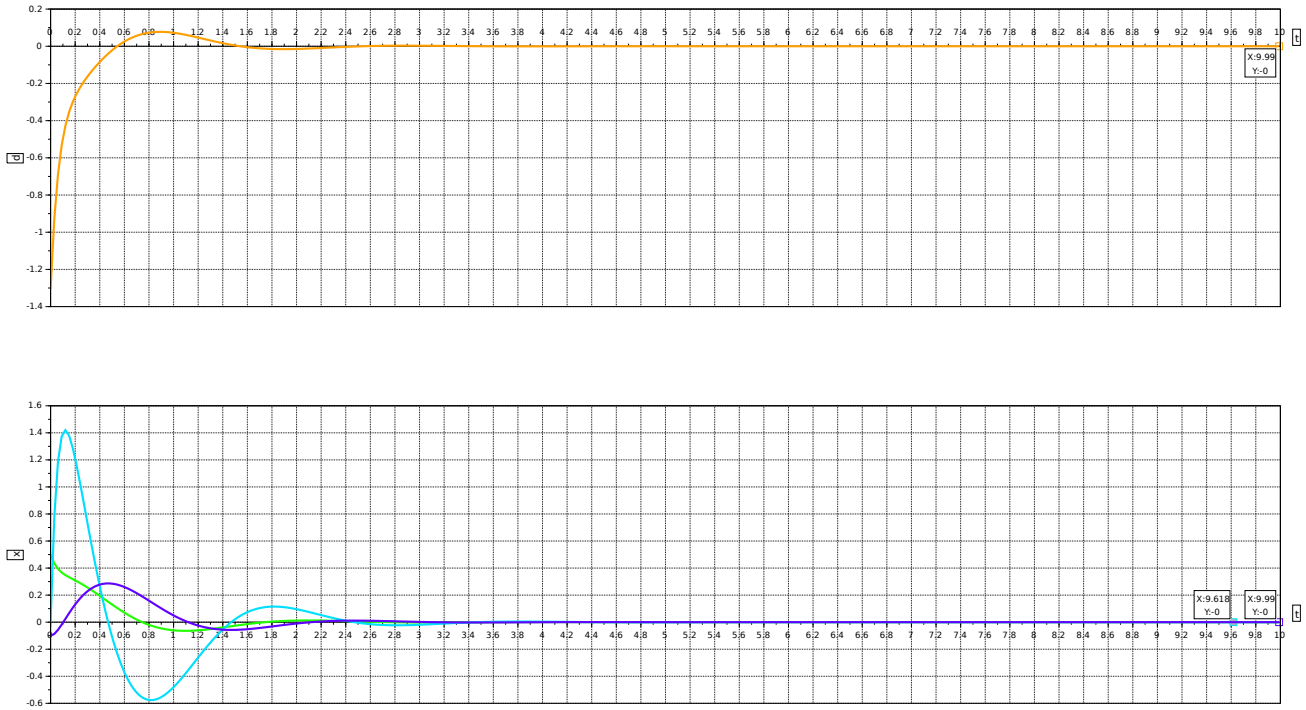
```
polos:
[[-16.0944+0.j    ]
 [ -1.6094+3.1416j]
 [ -1.6094-3.1416j]]
```

6 Implemente a alocação de polos no Simulink (MATLAB), Xcos (Scilab), ou similares. Assuma inicialmente que o controlador tem acesso ao vetor de estados completo. Inclua imagem(ns) do controlador construído.

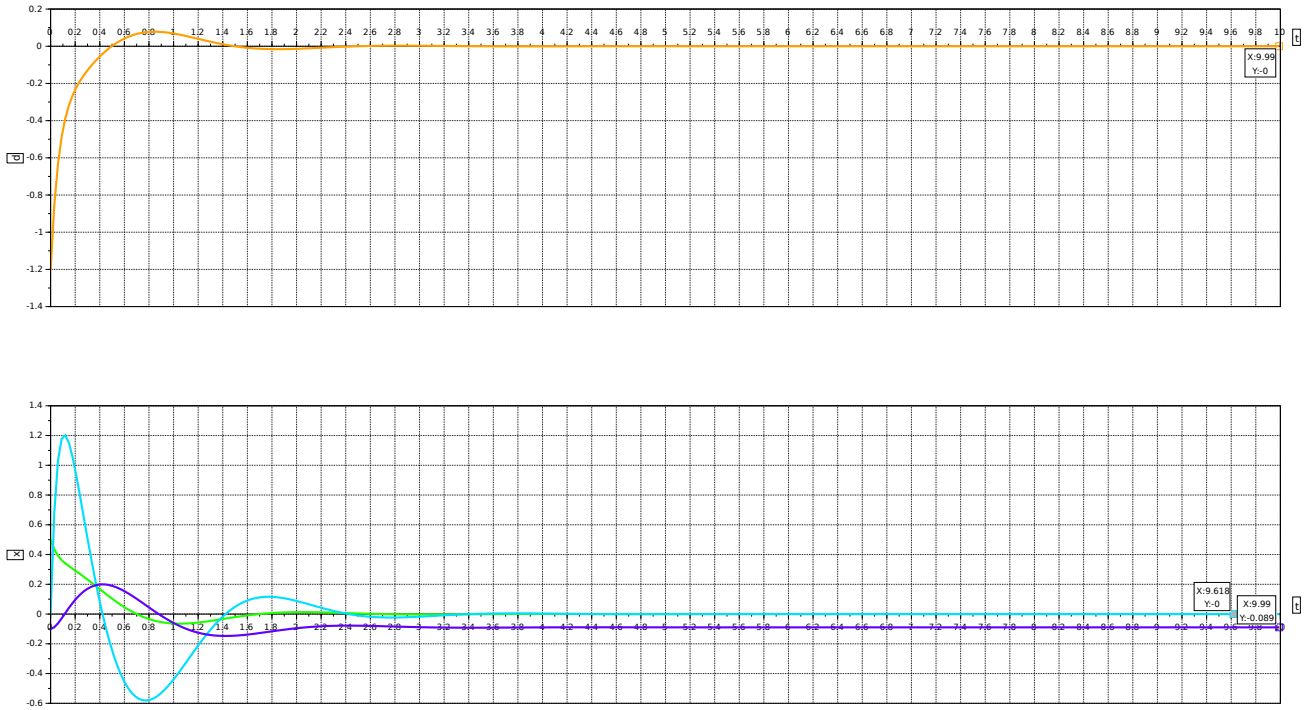
Diagrama do controlador (em amarelo) conectado à planta (em verde)



6.1 Simule o sistema com entrada nula



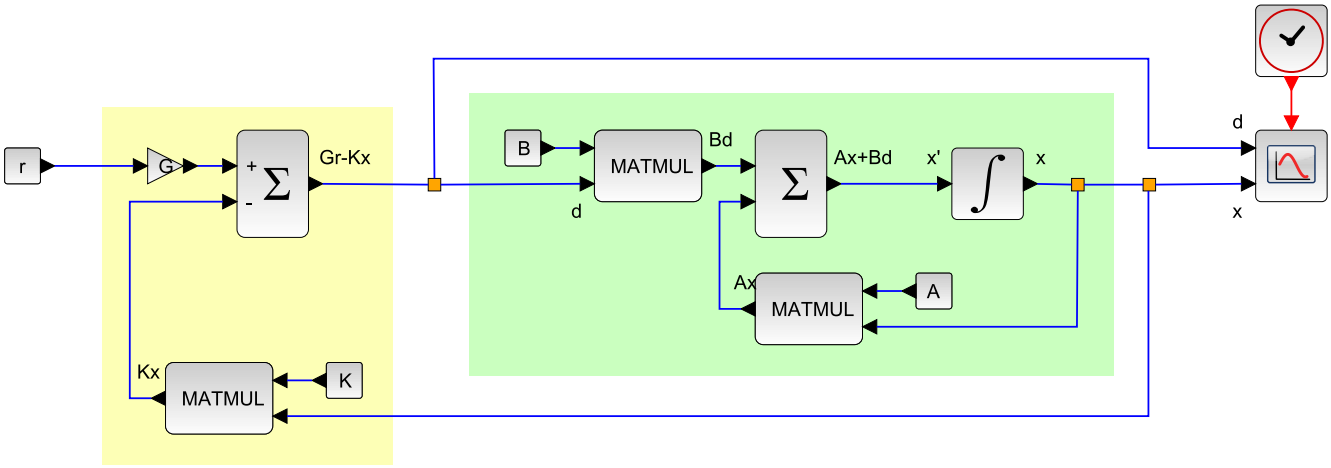
6.2 Simule o sistema com entrada degrau de 0.1 rad



O valor de sobressinal e o tempo de pico desejados não podem ser observados nos gráficos, pois as condições iniciais dominam o sistema durante o início da resposta transiente (os valores iniciais são muito maiores que a entrada). O sistema também manteve a estabilidade de α e q, mas agora com θ estável, não mais marginalmente estável. uma entrada degrau agora desloca θ, não q e α, para um degrau;

7 Altere controlador para que se transforme em um rastreador com erro ao degrau nulo em regime permanente. O sinal de entrada define o ângulo de arfagem θ desejado.

O modelo de rastreador é uma modificação simples do modelo anterior, adicionando apenas um ganho proporcional G à entrada de referência r. O diagrama completo pode ser visto na imagem abaixo, enquanto o cálculo do ganho é realizado em código python.

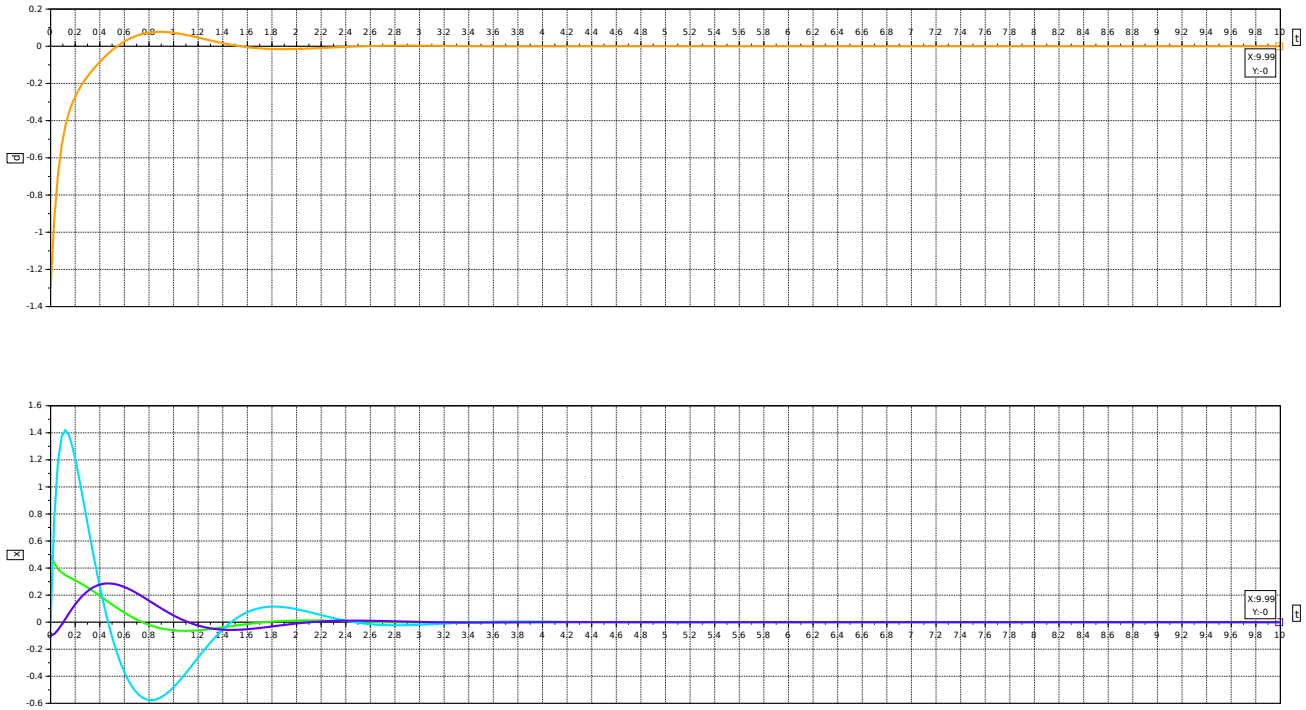


[8]:
```python
# Função para calcular o ganho proporcional
def gain_scale(k, a, b, c):
    """
    Calcula o ganho proporcional da referência para um rastreador com feedback completo

    Modelo:
    dx = Ax + Bu
    y  = Cx
    u = Gr - Kx

    Método:
    https://www.control.utoronto.ca/people/profs/kwong/ece410/2008/notes/chap5.pdf
    G = [K I] * [[A B], [C 0]]^(-1) * [0 I]'
    """

    o = np.zeros_like(c).T
    i = np.eye(c.shape[0])

    m1 = np.block([k, i])
    m2 = np.linalg.inv(np.block([[a, b], [c, 0 * i]]))
    m3 = np.block([[o], [i]])

    g = m1 @ m2 @ m3
    return g
```

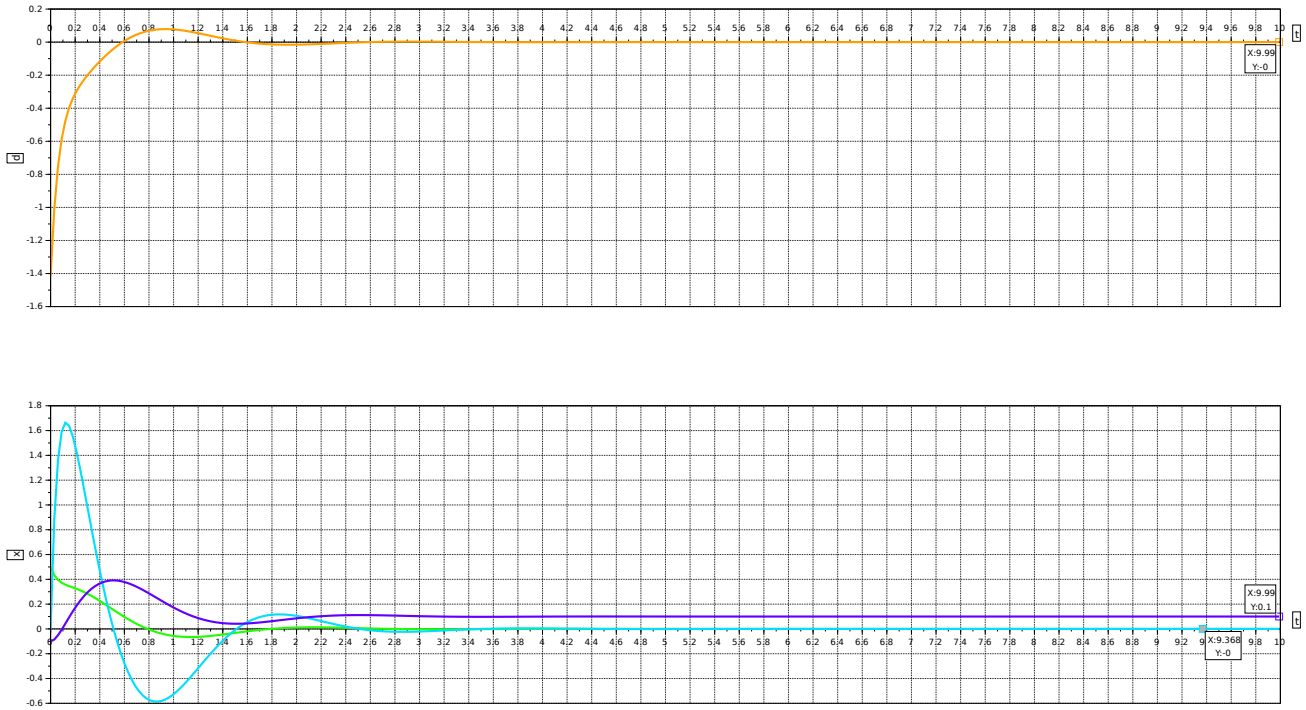
[9]:
```python
# Encontra o ganho proporcional
C = np.array([[0, 0, 1]])  # Rastreia theta
G = gain_scale(K, A, B, np.array([[0, 0, 1]]))
print("G:")
print(G)
```

```
G:
[[-1.1182]]
```

7.1 Simule o sistema com entrada nula



7.2 Simule o sistema com degrau de 0.1 rad



Conforme desejado, o estado θ agora rastreia a entrada r com erro nulo ao degrau

8 Projetar e simular dois reguladores diferentes projetados via LQR, listados abaixo.

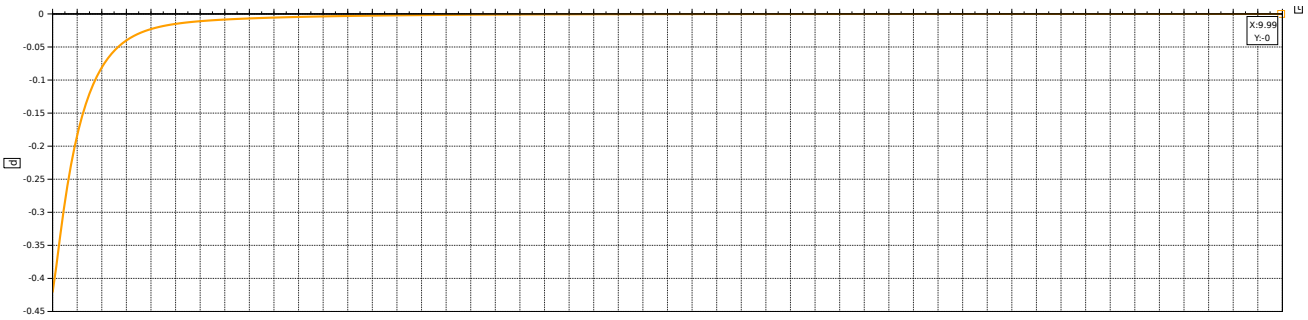
O modelo do controlador LQR é o mesmo do controlador por alocação de polos desenvolvido acima mudando apenas o método de escolha do ganho/polos.

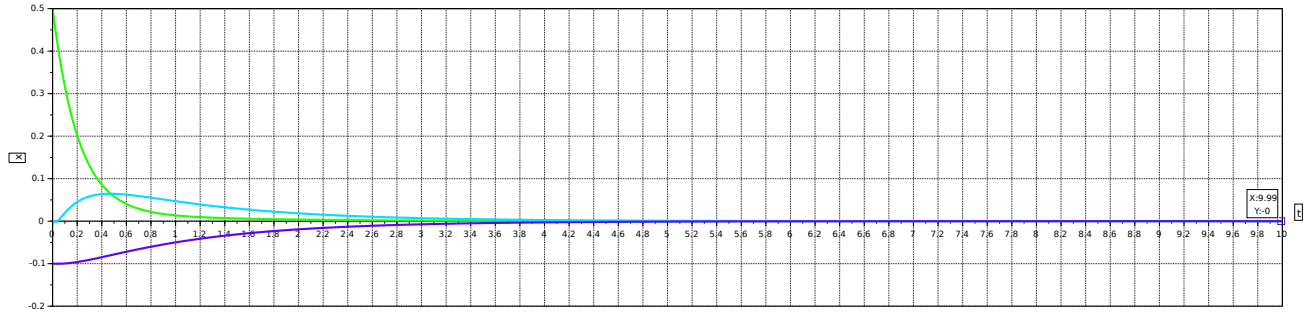
8.1 Estados e sinais de controle com pesos iguais

[10]:
```
# LQR com pesos iguais
Q1 = np.eye(3)
R1 = np.array([[1]])
K1, _, eig1 = control.lqr(A, B, Q1, R1)
# Calculado também o ganho proporcional, não necessário para a entrada nula
G1 = gain_scale(K1, A, B, C)

print("Q1:")
print(Q1)
print()

print("R1:")
print(R1)
print()

print("K1:")
print(K1)
print()

print("G1:")
print(G1)
print()

print("polos:")
print(np.reshape(eig1, (3, 1)))
```

```
Q1:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

R1:
[[1]]

K1:
[[ 0.6423 -0.9274 -1.    ]]

G1:
[[-1.]]

polos:
[[-38.4769+0.j]
 [ -4.8481+0.j]
 [ -0.9614+0.j]]
```

A resposta no tempo do sistema, mostrada abaixo, mostra que há uma resposta superamortecida. O pico de uso do atuador é de aproximadamente -0.46 rad.





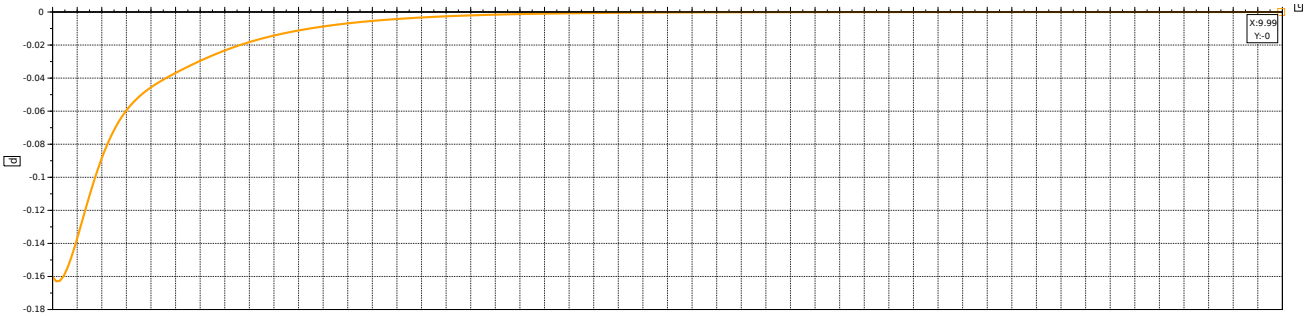
8.2 Um ajuste que poupa (reduz) o uso do atuador (profundor)

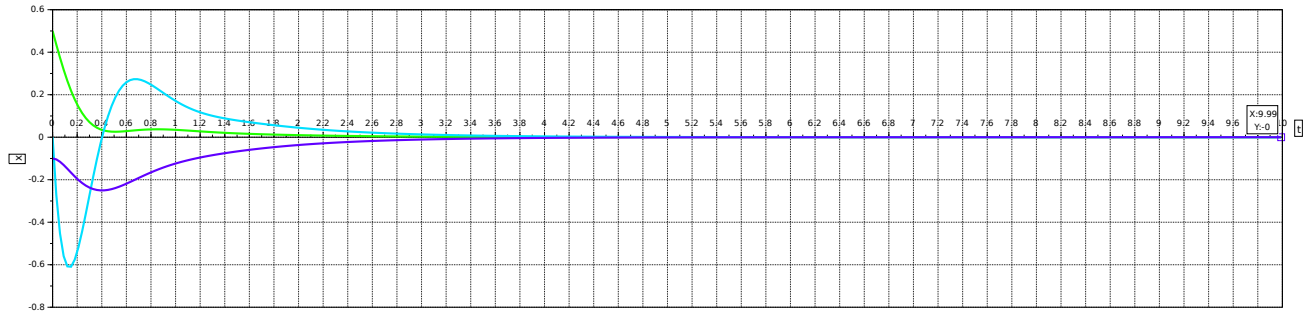
[11]:
```
# LQR com pesos variáveis, focando em baixo uso de atuador
# Optado por diminuir também a importância dos estados não rastreados
Q2 = np.array([[0.1, 0, 0], [0, 0.1, 0], [0, 0, 1]])
R2 = np.array([[10]])
K2, _, eig2 = control.lqr(A, B, Q2, R2)
G2 = gain_scale(K2, A, B, C)

print("Q2:")
print(Q2)
print()

print("R2:")
print(R2)
print()

print("K2:")
print(K2)
print()

print("G2:")
print(G2)
print()

print("polos:")
print(np.reshape(eig2, (3, 1)))
```

```
Q2:
[[0.1 0.  0. ]
 [0.  0.1 0. ]
 [0.  0.  1. ]]

R2:
[[10]]

K2:
[[ 0.2573 -0.0656 -0.3162]]
```

```
G2:
[[-0.3162]]

polos:
[[-4.5892+5.066j]
 [-4.5892-5.066j]
 [-1.2137+0.j    ]]
```

Conforme desejado, o segundo controlador apresenta menor uso do atuador (pico sai de -0.46 a -0.16). Entretanto, esse baixo uso do atuador tem como consequência adversa uma piora do comportamento transiente do sistema. θ e q iniciamnete se distanciam do valor estável, devido a forças da dinâmica passiva da planta superiores à força do atuador (α alto gera momento alto).





Lista 3

9 Determine a matriz de saída C para cada medida a seguir:

9.1 Ângulo de arfagem θ

Ângulo de arfagem é o terceiro estado do sistema.

[12]:
```
# y = theta
C_theta = np.array([[0, 0, 1]])
print("C_theta:")
print(C_theta)
```

```
C_theta:
[[0 0 1]]
```

9.2 Ângulo de ataque α

Ângulo de ataque é o primeiro estado do sistema.

[13]:
```
# y = alpha
C_alpha = np.array([[1, 0, 0]])
print("C_alpha:")
print(C_alpha)
```

```
C_alpha:
[[1 0 0]]
```

9.3 Direção de voo γ

Direção de voo não é um estado do sistema, mas pode ser calculado como a diferença entre ângulo de arfagem e ângulo de ataque.

[14]:
```
# y = gamma = theta - alpha
C_gamma = np.array([[-1, 0, 1]])
print("C_gamma:")
print(C_gamma)
```

```
C_gamma:
[[-1  0  1]]
```

10 Avalie a observabilidade para cada um dos sensores separadamente.

[15]:
```
# Vetoriza a operação
C_array = np.array([C_theta, C_alpha, C_gamma])
# Calcula matrizes de observabilidade
obs = np.block([[C_array], [c := C_array @ A], [c := c @ A]])
# Verifica observabilidade
# (posto da matriz de observabilidade igual a número de estados)
print(np.linalg.matrix_rank(obs) == np.shape(A)[0])
```

```
[ True False  True]
```

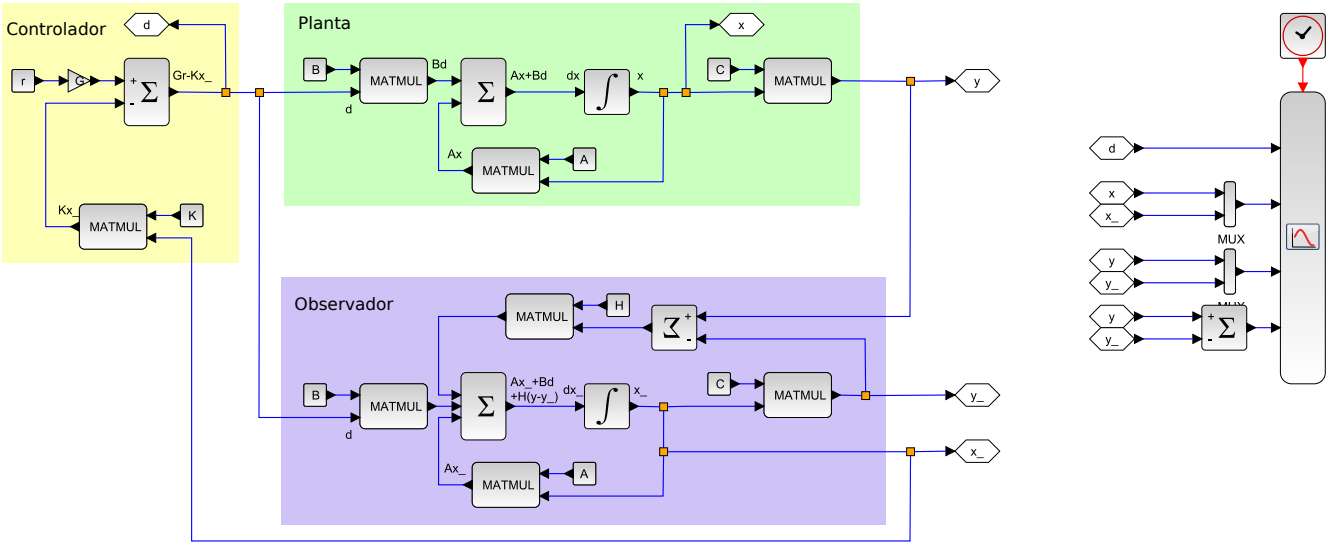
O sistema é observável para medições de ângulo de arfagem ou de direção do voo, mas não é plenamente observável com apenas medição do ângulo de ataque

11 Projete um observador de estados de ordem plena que possua um tempo de acomodação compatível com o controlador projetado na Questão 7. Considere que a saída do sistema é o ângulo de arfagem θ.

[16]:
```
# dx_ = A @ x_ + B @ d + H @ (y - y_)
# y_ = C * x_
# x0_ = 0

C = C_theta
# polos convergindo 2 vezes mais rápido que controlador
polos_observador = np.linalg.eigvals(A - B @ K).real * 2
# Ganho do observador
H = control.acker(A.T, C.T, polos_observador).T

print("H:")
print(H.T)
```

```
H:
[[ 20.1995 -61.3949  30.9364]]
```

12 Modele o observador de estados no Simulink. Altere o controlador da Questão 7 para utilizar as informações provindas do observador de estados. Mostre imagem(ns) do sistema com planta e observador de estados. Assuma que as condições iniciais do observador são nulas (o observador não conheceo estado inicial da planta). Inclua imagem(ns) do observador construído.



13 Simule o sistema nos seguintes casos:

Mostrar gráficos contendo

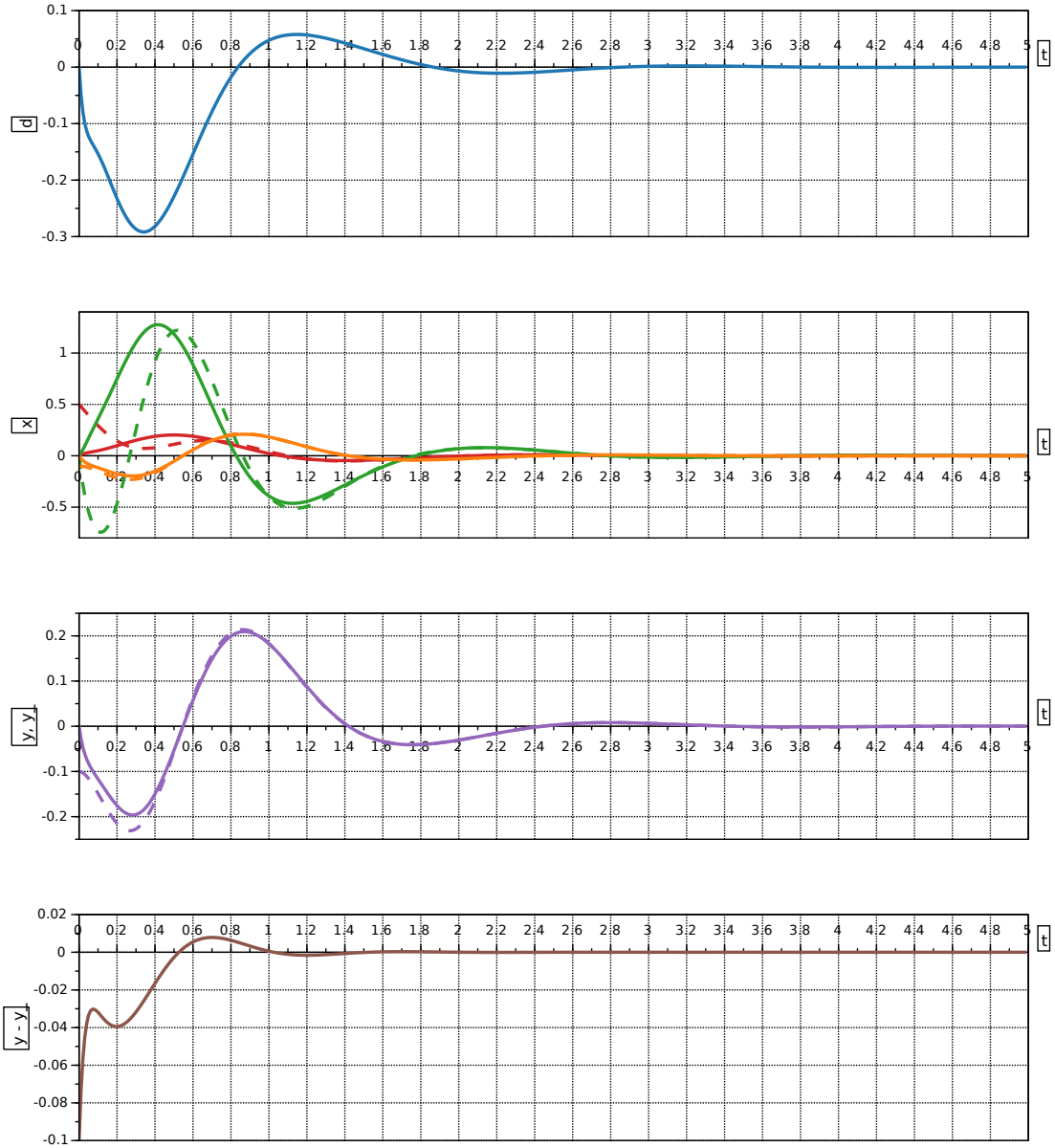
- um gráfico com sinal de controle δ
- o vetor de estados (ou seja, dos 3 estados)
- $y(t)$ e $\hat{y}(t)$ um mesmo gráfico
- $y(t) - \hat{y}(t)$

Discuta os resultados

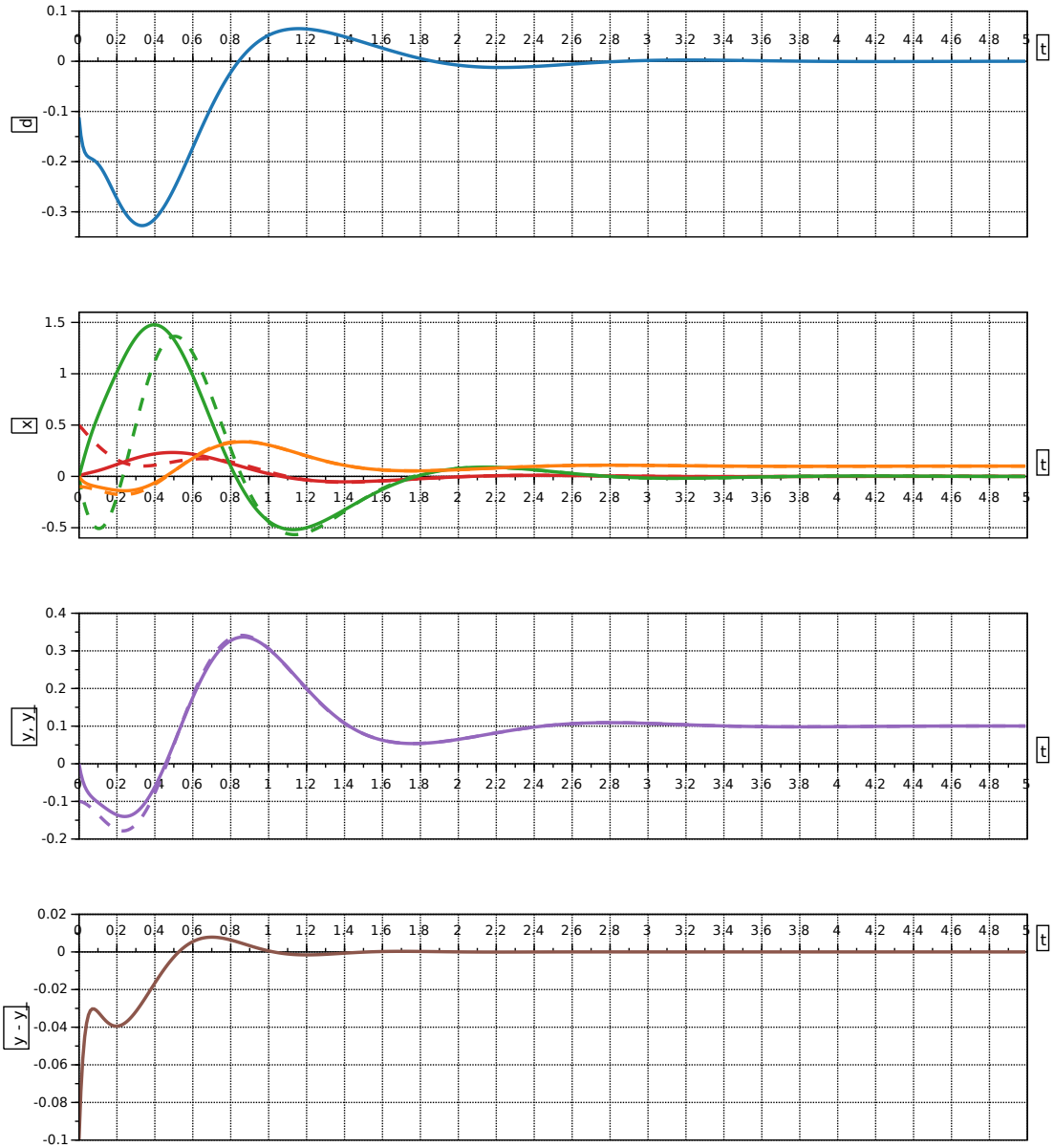
As figuras abaixo mostram os resultados da simulação do sistema. O sinal de controle é mostrado no primeiro gráfico em azul, os estados α, q, θ são mostrados no segundo gráfico em vermelho, verde e laranja, respectivamente. O sinal de saída é mostrado no terceiro gráfico em roxo, e o erro associado à saída é mostrado no quarto gráfico em marrom. Para os estados e a saída, as linhas pontilhadas indicam os valores "reais" (obtidos diretamente da planta), enquanto as linhas contínuas indicam os valores calculados pelo observador de estados.

Podemos ver que o observador converge em tempo adequado (projetado como metade do tempo de acomodação do sistema), e o sistema como um todo é controlado rapidamente. É importante notar que a acomodação do observador depende apenas do estado inicial, não variando com a entrada (pode ser visto a partir do gráfico do erro de saída, que é igual em ambas as simulações).

13.1 Entrada nula



13.2 Entrada degrau de 0.1 rad



14 Altere o controlador para incluir um canal integral. Simule o sistema nos seguintes casos:

Mostrar gráficos contendo

- um gráfico com sinal de controle δ
- o vetor de estados (ou seja, dos 3 estados)
- $y(t)$ e $\hat{y}(t)$ um mesmo gráfico
- $y(t) - \hat{y}(t)$

Discuta os resultados

Para adicionar o canal integrador, precisamos montar um sistema com vetor de estados aumentado. Queremos integrar a saída, de forma que nossas equações de estado aumentadas são descritas da seguinte maneira (com matriz **D** nula):

$$\dot{x_a} = r - y \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{x_a} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ -\mathbf{C} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ x_a \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ 0 \end{bmatrix} [u] + \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} [r]$$

$$[y] = [\mathbf{C} \quad 0] \begin{bmatrix} \mathbf{x} \\ x_a \end{bmatrix}$$

Esse sistema está na forma de espaço de estados, e podemos considerá-lo um sistema aumentado, utilizando as novas matrizes definidas:

[17]:
```
n_out = C.shape[0]
n_x = C.shape[1]
n_in = B.shape[1]

Aa = np.block([[A, np.zeros((n_x, n_out))], [-C, np.zeros((n_out, n_out))]])
print("Aa:")
print(Aa)
print()


Ba = np.block([[B], [np.zeros((n_out, n_in))]])
print("Ba:")
print(Ba)
print()


Ca = np.block([C, np.zeros((n_out, n_out))])
print("Ca:")
print(Ca)
print()
```

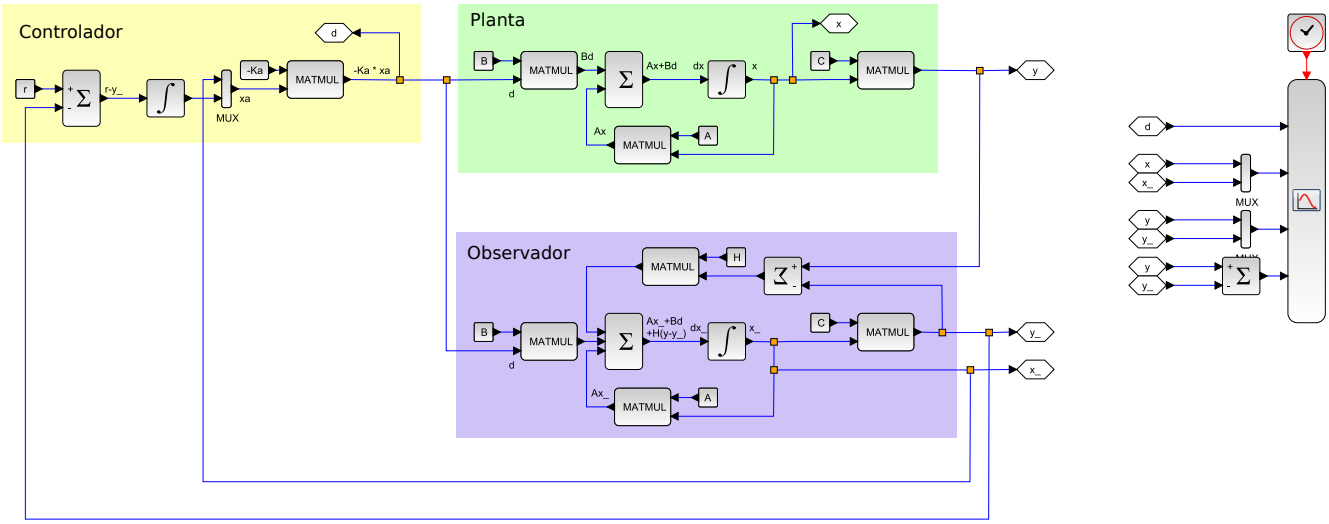
```
Aa:
[[ -4.1172   0.7781   0.       0.    ]
 [-33.8836  -3.5729   0.       0.    ]
 [  0.       1.       0.       0.    ]
 [  0.       0.      -1.       0.    ]]

Ba:
[[  0.5435]
 [-39.0847]
 [  0.    ]
 [  0.    ]]

Ca:
[[0. 0. 1. 0.]]
```

Com o novo sistema, precisamos alocar os polos do controlador novamente. Para manter as características de projeto e para facilitar as comparações com o sistema sem o canal integrado, os polos são alocados nos mesmos locais dos polos anteriores. O novo polo é alocado próximo ao polo mais distante do sistema anterior (não são sobrepostos para evitar erros numéricos).

[18]:
```
# Aloca polos no sistema extendido
polos_old = np.linalg.eigvals(A - B @ K)
polos_a = np.block([*polos_old, 1.01 * polos_old[0]])
Ka = control.place(Aa, Ba, polos_a)

print("Ka:")
print(Ka)
print()
print("polos:")
print(np.linalg.eigvals(Aa - Ba @ Ka))
```

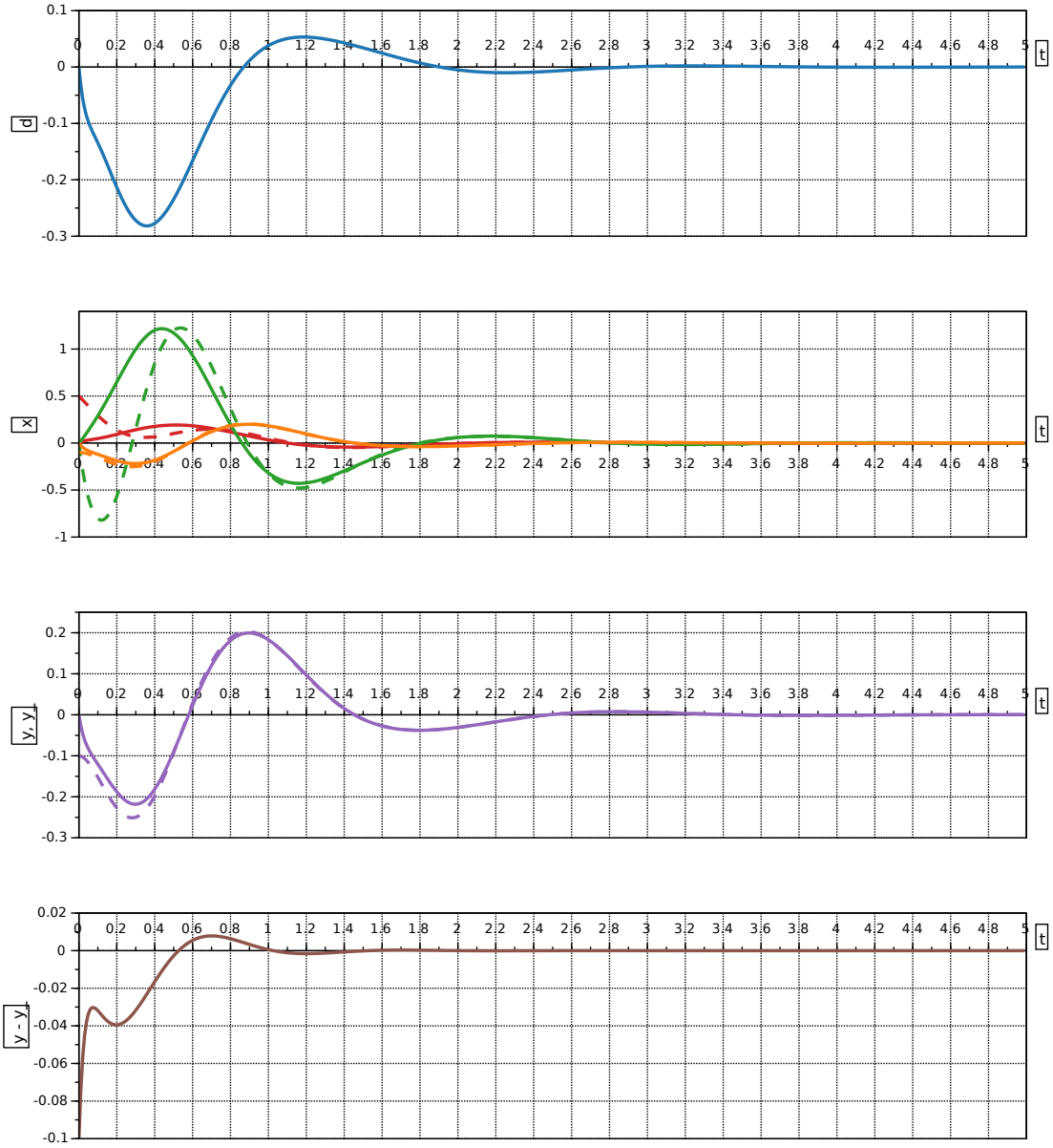
```
Ka:
[[-2.9929 -0.7549 -2.9819 18.1768]]

polos:
[-16.2553+0.j      -16.0944+0.j      -1.6094+3.1416j  -1.6094-3.1416j]
```

Para montar o diagrama de blocos do sistema, precisamos apenas modificar o controlador. A diferença entre a saída e o valor de referência é calculada e integrada antes do ganho, o vetor de estados aumentado é montado multiplicado pelo vetor de ganhos.



As figuras abaixo mostram os gráficos do sistema simulado. Os valores estão exibidos da mesma maneira que na questão anterior. Comparando ambas as respostas, podemos ver que a resposta não foi alterada significativamente, conforme desejado. A integração também não altera a performance do observador, já que o observador atua diretamente na planta original.

14.1 Entrada nula



14.2 Entrada degrau de 0.1 rad