UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

SEL0326- Controle de Sistemas Lineares

Controle LQR

Felipe Andrade Garcia Tommaselli (11800910)

Prof. Rodrigo Ramos

São Carlos
1 de dezembro de 2023

Sumário

Lista de Figuras **2**

1 Introdução **3**

2 Desenvolvimento **4**
2.1 Criação do Modelo nominal do grupo 5
2.2 Definição do funcional LQR 5
2.3 Cálculo do ganho ótimo K para o modelo nominal do grupo 6
2.4 Avaliação da otimalidade do controle fora das condições nominais 9
2.5 Extra: Otimização de Q e R 10

3 Conclusão **14**

Lista de Figuras

1 Definição das matrizes A e B 5
2 Definição das matrizes L e Q 6
3 Resolução da equação de Riccati 7
4 Código para simular o sistema em espaço de estados com x0 7
5 Resultado da resposta em malha aberta e fechada (LQR) 8
6 Simulação de J 8
7 100 ensaios variando ΔA 9
8 Plot 100 ensaios variando ΔA 9
9 Valores de J para 100 ensaios 10
10 Resposta para L alto 11
11 Resposta para Q alto 12
12 Resultado ótimo encontrado 13

1 Introdução

Tratando do escopo da disciplina, controlar sistemas no geral por meio de aproximações lineares é uma técnica amplamente utilizada no estado da arte. Considerando que em torno de um ponto x_e de equilíbrio, a Série de Taylor permite uma aproximação boa o suficiente do comportamento do sistema. Com isso, técnicas como o controle PID, alocação de polos e Controle LQR se consolidaram no controle desses sistemas.

O controle LQR (Linear Quadratic Regulator) é um tipo de controle ótimo que minimiza uma função de custo quadrática. Esse controlador é projetado para estabilizar um sistema linear minimizando a função de custo:

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{1}$$

Onde x é o vetor de estado, u é a entrada de controle, Q é a matriz de ponderação do estado e R a matriz de ponderação do controle. O controlador LQR calcula a entrada de controle ótima u^* que minimiza a função de custo. Essa entrada é dada por:

$$u^* = -Kx \tag{2}$$

Sendo K é a matriz de ganho, a qual é calculada resolvendo a equação de Riccati algébrica:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{3}$$

Onde P é a solução da equação de Riccati e a matriz de ganho LQR é:

$$K = R^{-1}B^T P \tag{4}$$

O LQR é um controlador de realimentação de estado, o que significa que a entrada de controle é uma função do vetor de estado. Esse controlador apresenta como característica permitir que o projetista opte entre performance e custo, os quais são otimizados por ele para o problema desejado. O presente trabalho apresentará um controlador LQR para um um sistema de potência do tipo Máquina versus Barramento Infinito em torno de uma condição particular de operação, a qual é instável em malha aberta.

2 Desenvolvimento

O sistema linear que representa o sistema é dado por:

$$\dot{x} = Ax + Bu, \quad x(0) = x_0$$

Para:

$$A = \begin{bmatrix} 0 & 376.9911 & 0 & 0 \\ -0.15685 & 0 & -0.0784 & 0 \\ -0.16725 & 0 & -0.46296 & 0.166667 \\ 1572.825 & 0 & -5416.98 & -100 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 100 \end{bmatrix}$$

Para os desenvolvimentos abaixo, foi utilizado "Python"como ambiente para simulação do sistema e do controlador [1]. Além disso, todos os códigos e desenvolvimentos estão disponíveis no formato de "Jupyter Notebook"no Repositório remoto no Github do discente, sendo possível acessar pelo navegador o código completo e algumas explicações mais a fundo do desenvolvimento [2].

[1] Devido a falta de diponibilidade do discente ao sistema operacional Windows para utilizar o "Matlab", opotou-se por utilizar "Python"no sistema operacional Linux. Visto que o discente já possui familiaridade com a linguagem, os resultados obtido foram exatamente os mesmos de fazer as simulações pelo Matlab.

[2] Github do projeto: `https://github.com/Felipe-Tommaselli/AdvancedControl/blob/main/T1_SisLinCtrl/T1_SisLinCtrl.ipynb`

2.1 Criação do Modelo nominal do grupo

O código de definição das matrizes está representado na figura 1.

Figura 1: Definição das matrizes A e B

```
# Matrix A
A = np.array([[0, 376.9911, 0, 0],
              [-0.15685, 0, -0.0784, 0],
              [-0.16725, 0, -0.46296, 0.166667],
              [1572.825, 0, -5416.98, -100]])

# Matrix B
B = np.array([[0],
              [0],
              [0],
              [10000]])

# Generate delta_A
delta_A = np.random.uniform(-0.05, 0.05, size=A.shape) * A

A = A + delta_A

A.shape, B.shape
```

A matriz ΔA foi criada com $\pm$ 5% dos valores de A gerados aleatoriamente, criando a matriz $A + \Delta A$.

2.2 Definição do funcional LQR

Para definir o funcional LQR, deve-se analisar as dimensões das matrizes A e B no cálculo do funcional. O controlador LQR é projetado para estabilizar um sistema linear minimizando a função de custo:

$$J = \int_0^\infty (x^T Q x + u^T R u) dt$$

Note que, para a multiplicação $x^T Q x$, a matriz Q deve ter a mesma dimensão da matriz A (4x4) para fornecer a mesma dimensão de multiplicação com x. Além disso, com $u^T R u$, sabendo que B não é uma matriz quadrada, para manter a multiplicidade correta de $u^T R u$, R deve ser uma matriz 1x1.

Os primeiros tetes conduzidos foram feitos com a definição das matriz identidades,

porém, posteriormente será apresentado uma resultado ótimo com base em estudados teóricos e práticos da aplicação do modelo.

Figura 2: Definição das matrizes L e Q

```
# Q e R matrizes diagonais e postivias

Q = np.diag([1, 1, 1, 1])
R = np.diag([1])
```

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Q = \begin{bmatrix} 1 \end{bmatrix}$$

2.3 Cálculo do ganho ótimo K para o modelo nominal do grupo

Considerando as condições:

- Modelo: $\dot{x} = (A + \Delta A)x + Bu, \;\; x(0) = x_0$
- Realimentação: $u = -Kx$
- Sistema em malha fechada: $\dot{x} = (A + \Delta A - BK)x$
- Condição inicial: $x_0 = [0, 0, 0, 0.1]^T$

A matriz K pode ser encontrado pela equação de Riccati:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0$$

Figura 3: Resolução da equação de Riccati

```
# a) solve the algebraic Riccati equation to obtain the LQR gain matr
ix K
K, S, P = matlab.lqr(A, B, Q, R)

# create the closed-loop system with the LQR controller
Ac = A - B @ K

sys = matlab.ss(A, B, np.eye(4), 0)
sys_lqr = matlab.ss(Ac, B, np.eye(4), 0)
```

A matriz K encontrada foi:

$$K = [0.558,\ -2.117,\ -0.015,\ 0.991]$$

A partir disso, pode-se obter J por:

$$K = R^{-1}(B^T P)$$

$$A^T + PA - (PB)R^{-1}(B^T P) + Q = -\dot{P}$$

Como a função LQR retorna também a solução P, é possível calcular J a partir disso.

$$J = 9.901 \cdot 10^{-7}$$

Agora, para plotar os gráficos referentes a resposta do sistema para as condições iniciais x_0 dadas é necessário criar o espaço de estados com as matrizes A e B. Essa representação pode ser obtido como na figura 4 abaixo:

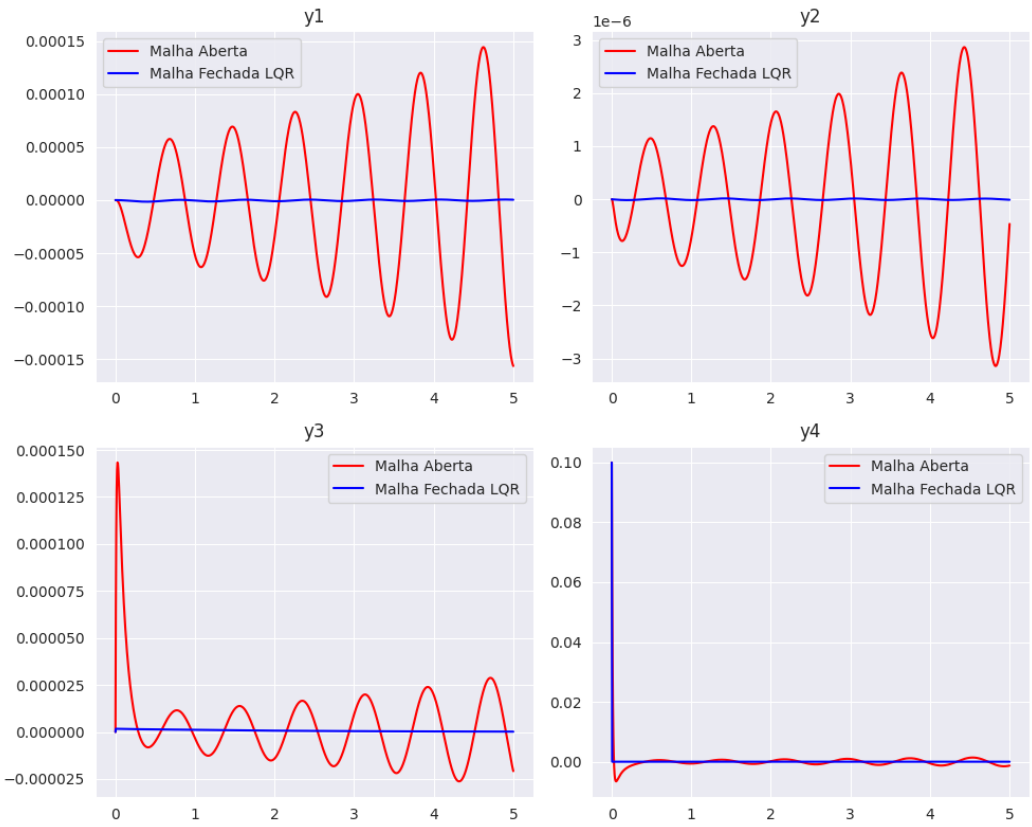
Figura 4: Código para simular o sistema em espaço de estados com x0

```
# b)
x0 = np.array([[0], [0], [0], [0.1]])

# simulate the system with the LQR controller
t = np.linspace(0, 5, 1000)
y, t, x = matlab.lsim(sys, 0 * t, t, x0)
ylqr, t, xlqr = matlab.lsim(sys_lqr, 0 * t, t, x0)
```

Os resultados podem ser avaliados pelas figuras 5 abaixo.

Figura 5: Resultado da resposta em malha aberta e fechada (LQR)



Perceba que o sistema é instável em malha aberta e estável em malha fechada. Além disso, Para fins de comparação, o valor de J foi encontrado pela expressão completa em código, como na imagem 6 abaixo.

Figura 6: Simulação de J

```
# Calculate J (simulation)
ylqr = ylqr.T
Ji_array = np.array([])  # Initialize an empty NumPy array to store the values of Ji

for j in range(len(ylqr)):
    Ji = ylqr[:, j].T @ Q @ ylqr[:, j] + (-K @ ylqr[:, j]).T @ R @ (-K @ ylqr[:, j])
    Ji_array = np.append(Ji_array, Ji)

print(np.trapz(Ji_array) * dt)
```

Com resultado:

$$J = 4.48 \cdot 10^{-7}$$

Note que a ordem de grandeza do funcional J calculado anteriormente pela equação de Ricatti e agora simulado pela expressão aproximado da integral na soma de Rienman

está na mesma ordem de grandeza (10^{-7}), mostrando coerência do resultado obtido.

2.4 Avaliação da otimalidade do controle fora das condições nominais

Para essa seção, 100 ensaios foram realizados com diferentes valores de ΔA aleatórios.

Figura 7: 100 ensaios variando ΔA

```
# Set the maximum number of tests
MAX = 100

# Create an empty list to store the values of the functional J
Jp_list = []
Ji = np.array([])
Ji_array = np.array([])

# Loop through the number of tests
for i in range(MAX):
    # Generate new random values for delta_A
    delta_A = np.random.uniform(-0.05, 0.05, size=A.shape) * A

    # Update the matrix A with the new delta_A values
    A_new = A + delta_A

    # Perform the desired calculations or simulations using the updated A_new
    K, S, P = matlab.lqr(A_new, B, Q, R)

    # create the closed-loop system with the LQR controller
    Ac = A_new - B @ K
    sys_lqr = matlab.ss(A_new, B, np.eye(4), 0)

    # simulate the system with the LQR controller
    ylqr, t, xlqr = matlab.lsim(sys_lqr, 0 * t, t, x0)

    # Calculate J (simulation)
    ylqr = ylqr.T
    Ji_temp = 0
    Ji = np.array([])
    for j in range(len(ylqr)):
        Ji_temp = ylqr[:, j].T @ Q @ ylqr[:, j] + (-K @ ylqr[:, j]).T @ R @ (-K @ ylqr[:, j])
        Ji = np.append(Ji, Ji_temp)

    Ji_array = np.append(Ji_array, np.trapz(Ji) * dt)
```

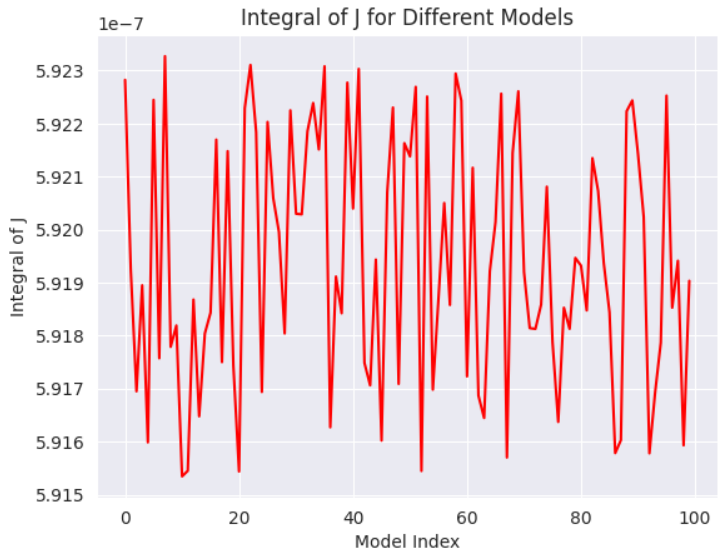
O resultado obtido de J pode ser visualizado na figura 9 pelo código da figura 8.

Figura 8: Plot 100 ensaios variando ΔA

```
# Create a list of model indices
model_indices = list(range(MAX))

# Plot the integral of J for each model
plt.plot(model_indices, Ji_array, label='Integral of J', color='red')
plt.xlabel('Model Index')
plt.ylabel('Integral of J')
plt.title('Integral of J for Different Models')
plt.show()
```

Figura 9: Valores de J para 100 ensaios



Vale ressaltar que as incertezas afetam o desempenho do modelo, uma vez que o valor de J variou a partir do valor mínimo de $5.915 \cdot 10^{-7}$ com a variância vista no gráfico da figura 9 dependendo do modelo utilizado. Além disso, o valor médio de J foi de $5.91 \cdot 10^{-7}$.

A partir disso fica evidente a teoria de minimização do funcional J, a qual já era esperado que fosse intimamente ligado com o modelo do sistema em questão, cabendo ao projetista analisar as condições de performance e custo desejadas.

2.5 Extra: Otimização de Q e R

Sabendo que as matriz Q e R representam as matrizes de peso para performance e custo em um sistema sistema de potência do tipo Máquina versus Barramento Infinito, uma análise do impacto dessas matrizes foi feita para avaliar o otimalidade do controle.

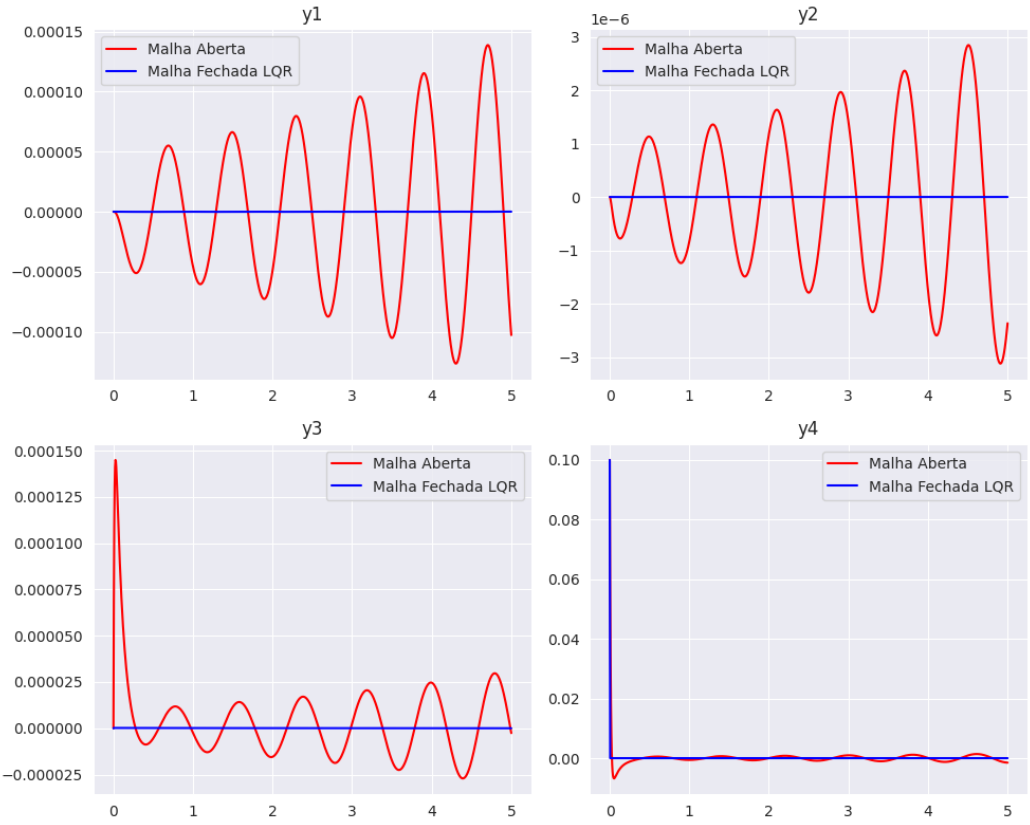
Nos resultados acima utilizou-se matrizes identidades para Q e R, contudo, caso seja de interesse do projetista ponderar o desempenho do projeto, a matriz Q acessa os estados do sistema e, portanto, caso tenha valores altos irá valorizar o desempenho crítico deles. Com isso, tomando:

$$L = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix}$$

$$Q = \begin{bmatrix} 1 \end{bmatrix}$$

Têm-se como resultado a figura 10.

Figura 10: Resposta para L alto



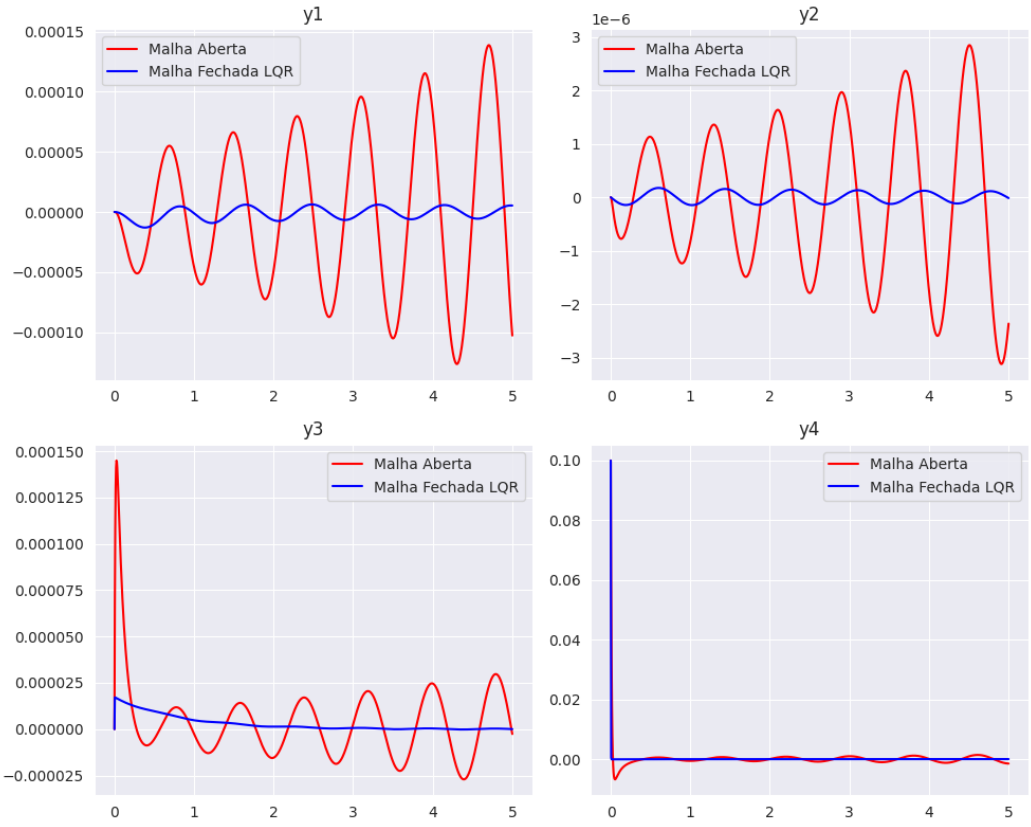
Com isso, é perceptível que a performance do sistema em termos de responsividade do controlador é extremamente efetiva. Agora, ponderando pelo termo Q que está ligado com as entradas e, portanto, representa o custo do sistema, têm-se:

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Q = \begin{bmatrix} 100 \end{bmatrix}$$

Têm-se como resultado a figura 11.

Figura 11: Resposta para Q alto



Onde é possível avaliar um comportamento diferente do último teste, ressaltando que a performance do sistema diminuiu.

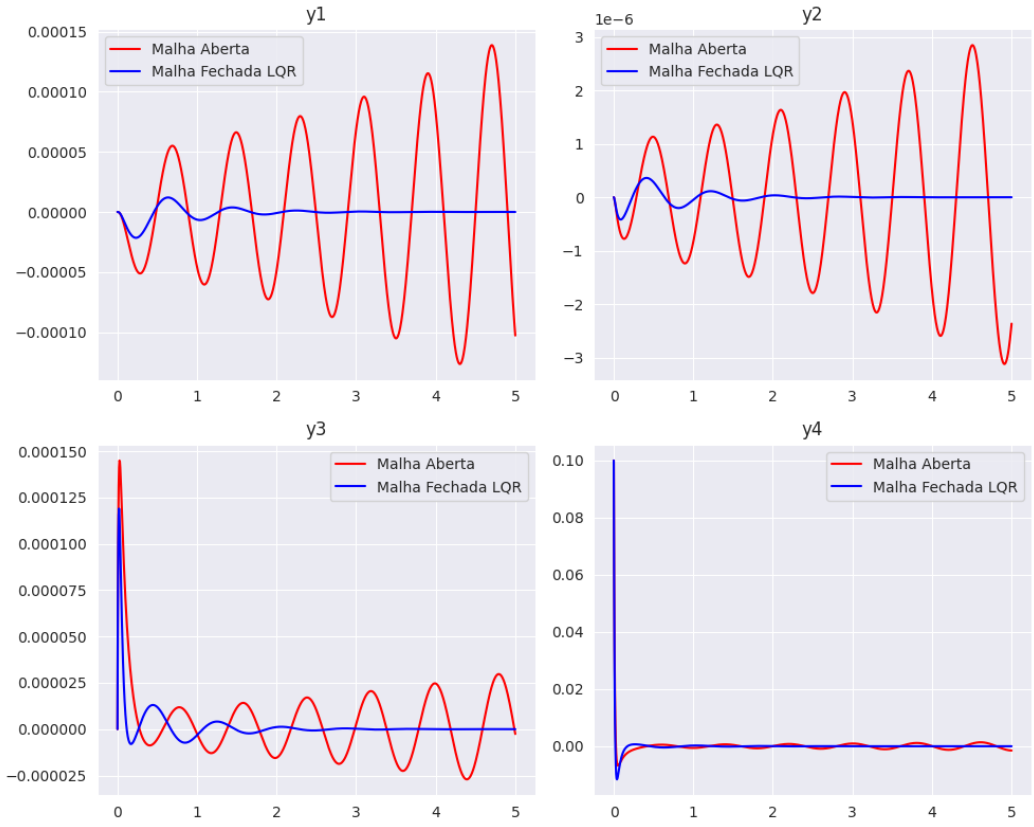
Na literatura, é possível encontrar algumas referências sobre controle ótimo para problemas de Máquina versus Barramento Infinito. Essa teoria aliada com alguns testes empíricos podem ser refletidos nas seguintes funções Q e R:

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$Q = \begin{bmatrix} 1 \end{bmatrix}$$

Que produzem como resultado ótimo a figura 12.

Figura 12: Resultado ótimo encontrado



3 Conclusão

O controle LQR representa um avanço significativo no campo de Controle de Sistemas Lineares, empregando ferramentas matemáticas sofisticadas para calcular um ganho ótimo por meio de uma função custo. A presença de uma literatura sólida, composta por livros, artigos e bibliotecas em MATLAB e Python, facilita a implementação prática do controle, tornando-o acessível para os engenheiros e pesquisadores.

A compreensão da intuição por trás do controle LQR é crucial, destacando-se como uma ponderação entre desempenho e energia de atuação. Ao equilibrar esses fatores, o controle visa alcançar resultados desejados pela análise dos requisitos de projeto do projetista.

Além disso, por meio da matriz de incerteza ΔA foi possível analisar de forma prática o impacto de mudanças no sistema na atuação do controlador para estabilização do sistema.

Em suma, o trabalho contribui significativamente para o avanço dos Tópicos de Controle Linear, oferecendo *insights* valiosos para pesquisadores, engenheiros e estudantes interessados no campo, proporcionando uma abordagem prática e teórica para a implementação eficaz do controle LQR.