UNIVERSIDADE FEDERAL DE ALAGOAS

Instituto de Computação – IC

Engenharia de Computação

Sistemas de Controle 2

IMPLEMENTAÇÃO DE UM SERVO-CONTROLADOR DO TIPO 0

Aldemir Melo Rocha Filho

Sandoval da Silva Almeida Junior

Tayco Murilo Santos Rodrigues

Maceió – 2022

ÍNDICE

1. Introdução

Este relatório tem como objetivo apresentar a utilização de um servo-controlador do tipo 0 para que o sistema elétrico de ordem três analisado no relatório anterior apresente o comportamento desejado em malha fechada. O controlador em questão foi implementado em linguagem $Python$ com auxílio de módulos presentes na biblioteca $control$ e $numpy$.

O link para acesso ao notebook com todos os algoritmos utilizados pode ser encontrado nos anexos deste documento.

2. Dinâmica do sistema em Espaço de Estado

Primeiro é preciso partir da representação geral de um sistema modelado em Espaço de Estados:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

Temos que na relação $\dot{x} = Ax + Bu$, o elemento Ax define a dinâmica do sistema, uma vez que os autovalores de A definem os polos do sistema, e o elemento Bu define como o sistema responde às entradas.

Dessa forma, para que um controlador modifique a dinâmica de um sistema é preciso que o mesmo modifique a matriz A e para que um controlador modifique como o sistema responde às entradas é preciso modificar a matriz B. A mudança da dinâmica do sistema pode ser feita realocando os polos, e a mudança da resposta do sistema pode ser feita recalculando a entrada do sistema a partir de um valor de erro observado na saída y.

3. Servo-Controlador do tipo 0

O controlador escolhido para realizar as tarefas descritas no item **2** pode ser observado abaixo:

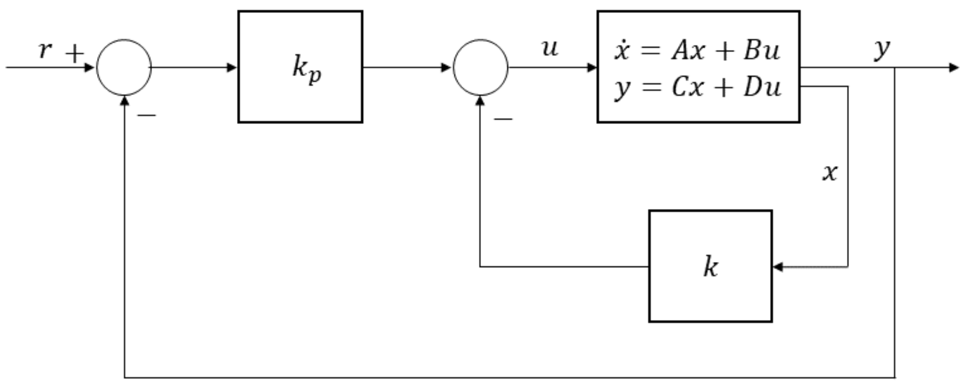

Imagem 3.1: Servo-Controlado do tipo 0

Observando o diagrama de blocos presente na **Imagem 3.1**, podemos perceber que em malha fechada a entrada do sistema é do seguinte formato:

$$u = r \cdot k_p - k \cdot x$$

De forma que a parcela $r \cdot k_p$ representa o sinal referência (r), como por exemplo um sinal degrau, multiplicado por um coeficiente de erro (k_p) e a parcela $k \cdot x$, a qual representa o vetor de estados (x) multiplicado por uma matriz de ganho (k). Dessa forma, temos que em nosso sistema, a relação $\dot{x} = Ax + Bu$ pode ser escrita da seguinte forma:

$$\dot{x} = Ax + Bu \rightarrow \dot{x} = Ax + B\left(r \cdot k_p - k \cdot x\right) \rightarrow \dot{x} = Ax - Bkx + Brk_p$$

Logo:

$$eq\ 3.1\text{:}\ \dot{x} = Ax + Bu = (A - Bk)x + \left(Bk_p\right)r$$

Dessa forma, o sistema apresentado na **Imagem 3.1** consegue garantir tanto a posição dos polos quanto o cálculo da nova entrada do sistema para compensar o erro de regime.

3.1. Matriz de ganho

A matriz de ganho k, como o nome indica, é uma matriz utilizada para realimentar a entrada do sistema com o objetivo de garantir um ganho no vetor de estados x e dessa forma modificar a dinâmica do sistema e consequentemente modificar a posição de seus polos. Os valores presentes em k são calculados partindo dos polos desejados para o sistema.

3.2. Coeficiente de erro K_p

O coeficiente de erro k_p é o valor utilizado para recalcular a entrada do sistema partindo do erro apresentado em y em comparação com o sinal de referencia r.

4. Dinâmica do sistema em Malha Aberta

4.1. Resposta temporal do sistema

A dinâmica do sistema em malha aberta pode ser observada alimentando o mesmo com um sinal de entrada referência e em seguida observando sua saída.

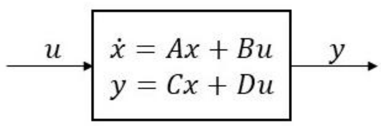

Imagem 4.1: Sistema em malha aberta

Primeiro partimos da montagem do sistema em malha aberta em um formato que o restante das implementações consiga interpretá-lo. O trecho de código utilizado pode ser observado abaixo:

```
from control import matlab

A = [[-1,1,0],[-1,0,-1],[0,1,0]]
B =[[0],[1],[0]]
C = [0,0,1]
D = [0]

sys_open = matlab.ss(A,B,C,D)
```

Imagem 4.2: Trecho de código utilizado para montar o sistema

Em seguida, precisamos montar o *array* que contem nosso sinal de entrada, para fins didáticos o sinal de entrada escolhido foi um degrau que varia em 3 valores diferentes com o passar o tempo. O trecho de código utilizado pode ser observado abaixo:

```
import numpy as np

tempo = np.arange(0,150,0.01) ##Vetor de tempo(150 segundos)

#Os valores do degrau mudam a cada 50 segundos
u1 = np.full(int(len(tempo)/3), 1) #Degrau unitario
u2 = np.full(int(len(tempo)/3), 3) #Degrau em 3
u3 = np.full(int(len(tempo)/3), 5) #Degrau em 5

u = np.concatenate((u1, u2, u3)) #Entrada com degrais de diferentes
    valores
```

Imagem 4.3: Trecho de código utilizado para montar os sinais de entrada

Por fim, partindo do vetor de tempo e do vetor que representa nosso sinal de entrada, podemos fazer uso do método *control.matlab.lsim()* para obtermos a resposta do sistema. O método em questão recebe como parâmetro o sistema, que está representado pela variável chamada *sys_open*, o sinal de entrada e o vetor de tempo. O trecho de código utilizado para se obter a resposta do sistema e gerar o *plot* pode ser observado abaixo:

```
from control import matlab

y_o, t_o, x_o = matlab.lsim(sys_open, u, tempo) #Resposta

##Sistema em malha aberta
plt.plot(t_o, y_o,'black')
plt.grid(alpha = 0.5)
plt.title('SISTEMA EM MALHA ABERTA')
plt.ylabel('Amplitude')
plt.xlabel('Tempo(s)')
plt.show()
```

Imagem 4.4: Trecho de código utilizado para se obter e plotar a resposta do sistema

A resposta do sistema obtida pode ser observada abaixo:

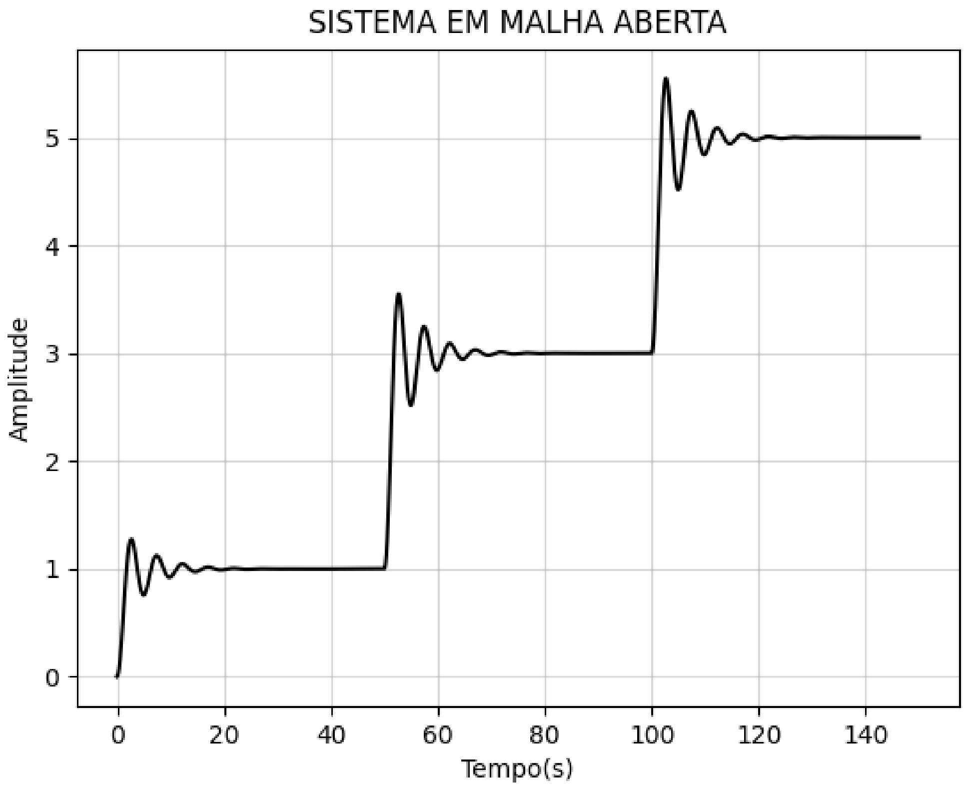

Imagem 4.5: Resposta do sistema em malha aberta

4.2. Polos do sistema em malha aberta

Como dito no tópico **2** os polos do sistema podem ser obtidos partindo dos autovalores da matriz A. Dessa forma, foi feito o uso do método $matlab.eigvals(\,)$. O trecho de código utilizado e as saídas obtidas podem ser observados abaixo:

```
##Polos do sistema em Malha aberta
print('POLOS DO SISTEMA EM  MALHA ABERTA:')
print(eigvals(A))
```

Imagem 4.6: Trecho de código utilizado para se obter os polos do sistema

```
POLOS DO SISTEMA EM  MALHA ABERTA:
[-0.56984029+0.j  -0.21507985+1.30714128j  -0.21507985-1.30714128j]
```

Imagem 4.7: Polos do sistema em malha aberta

Dessa forma temos que os polos do sistema em malha aberta são aproximadamente os seguintes:

$$Polos_{Malha\ aberta} = \begin{cases} p_1 \cong -0.5698 \\ p_2 \cong -0.2151 + 1.3071j \\ p_3 \cong -0.2151 - 1.3071j \end{cases}$$

5. Implementação dos Blocos do Servo-Controlador tipo 0

5.1. Polos desejados para o sistema e matriz de ganho $\boldsymbol{K}$

Os polos escolhidos para que o sistema em malha fechada apresente um comportamento mais rápido e com um menor sobressinal foram os seguintes:

$$Polos_{desejados} = \begin{cases} p_1 = -50 \\ p_2 = -1 + 0.1j \\ p_3 = -1 - 0.1j \end{cases}$$

Agora, partindo dos polos desejados é preciso calcular a matriz de ganho k a fim de se garantir a dinâmica do sistema. Assim, foi utilizado o método $control.matlab.place(\,)$ que recebe como parâmetro as matrizes A e B do sistema e o vetor com os *polos* desejados e retorna a matriz de ganho k. O trecho de código utilizado e a saída obtida podem ser observados abaixo:

```
from control import matlab

P = [-50, -1+0.1j, -1-0.1j] ## Polos desejados para o sistema
K = matlab.place(A,B,P) ##Matriz de ganho

print('MATRIZ DE GANHO PARA OS POLOS DESEJADOS: ')
print(K)
```

Imagem 5.1: Polos desejados para o sistema e cálculo da matriz de ganho

```
MATRIZ DE GANHO PARA OS POLOS DESEJADOS:
[[-1.49 51 49.5]]
```

Imagem 5.2: Matriz de ganho $\boldsymbol{k}$ para se obter a dinâmica desejada

Portanto, partindo da **Imagem 4.7**, temos que a matriz de ganho k é do seguinte formato:

$$k = (-1{,}49 \quad 51 \quad 49{,}5)$$

5.2. Matriz $\boldsymbol{A}$ em malha fechada

Partindo do diagrama de blocos apresentado na **Imagem 3.1** e da eq 3.1 temos que em malha fechada a matriz A do sistema vai assumir o seguinte formato:

$$A_{cl} = (A - Bk)$$

Dessa forma, podemos partir das matrizes A e B do sistema em malha aberta em conjunto de k, que foi calculado no passo anterior, para determinarmos A_{cl}. O trecho de código utilizado para este fim e a saída obtida podem ser observados abaixo:

```
Acl = (A - (np.dot(B,K))) ##Acl = (A - Bk)

print('MATRIZ A EM MALHA FECHADA: ')
print(Acl)
```

Imagem 5.3: Cálculo da matriz A em malha fechada

```
MATRIZ A EM MALHA FECHADA:
[[ -1.     1.     0.   ]
 [  0.49 -51.   -50.5 ]
 [  0.     1.     0.   ]]
```

Imagem 5.4: Matriz A do sistema em malha fechada

Portanto, partindo da **Imagem 5.4**, temos que a matriz A_{cl} do sistema em malha fechada é do seguinte formato:

$$A_{cl} = (A - Bk) = \begin{pmatrix} -1 & 1 & 0 \\ 0.49 & -51 & -50.5 \\ 0 & 1 & 0 \end{pmatrix}$$

5.2.1. Polos do sistema em malha fechada

Como dito no tópico **2** os polos do sistema podem ser obtidos partindo dos autovalores da matriz A_{cl}. Dessa forma, foi feito o uso do método $matlab.eigvals(\,)$. O trecho de código utilizado e as saídas obtidas podem ser observados abaixo:

```
##Polos do sistema em Malha Fechada
print('POLOS DO SISTEMA EM  MALHA FECHADA:')
print(eigvals(Acl))
```

Imagem 5.5: Trecho de código utilizado para se obter os polos do sistema

```
POLOS DO SISTEMA EM  MALHA FECHADA:
[-50.+0.j    -1.+0.1j   -1.-0.1j]
```

Imagem 5.6: Polos do sistema em malha fechada

Partindo da **Imagem 5.6** podemos observar que em malha fechada os polos do sistema foram para as posições desejadas, consequentemente a dinâmica do sistema foi modificada.

5.2.2. Resposta temporal do sistema com os polos realocados

Podemos repetir os paços apresentados no item **4** para obter a resposta do sistema com polos nas posições desejadas. O trecho de código utilizado e a saída obtida podem ser observados abaixo:

```
from control import matlab

sys_close = matlab.ss(Acl,B,C,D) ##Sistema em malha fechada
y_c, t_c, x_c = matlab.lsim(sys_close , u, tempo)#Resposta

##Resposta - Sistema em malha fechada
plt.plot(t_c, y_c,'black')
plt.grid(alpha = 0.5)
plt.title('SISTEMA COM POLOS REALOCADOS')
plt.ylabel('Amplitude')
plt.xlabel('Tempo(s)')
plt.show()
```

Imagem 5.7: Trecho de código utilizado para se obter e plotar a resposta do sistema

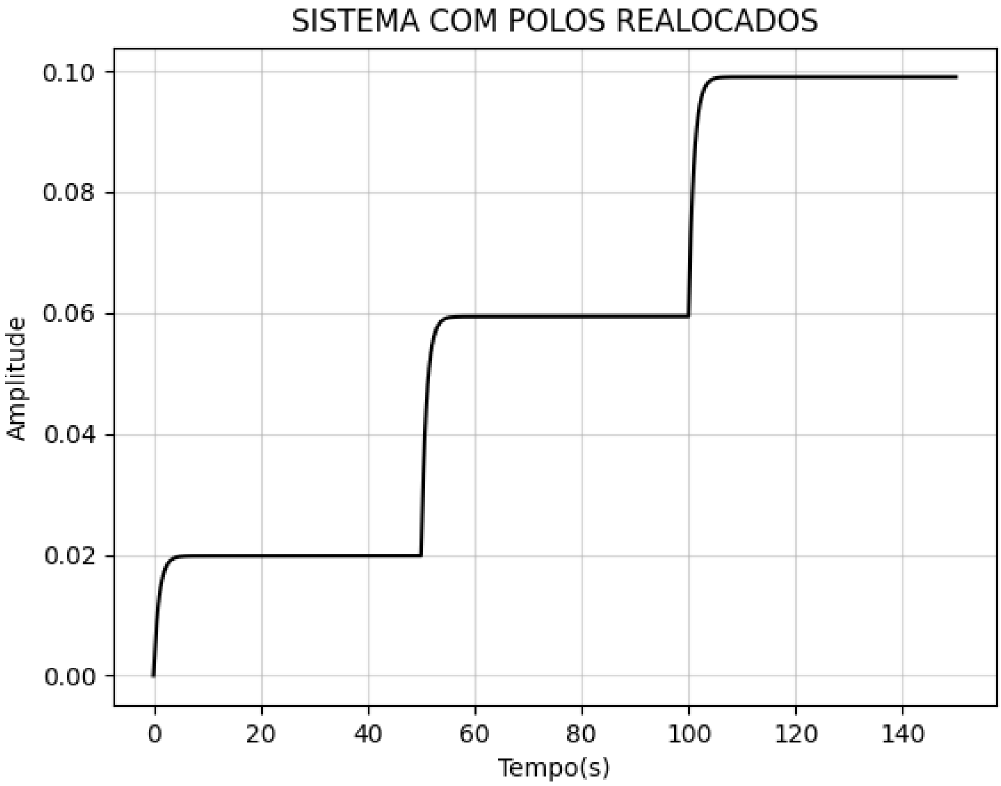

Imagem 5.6: Resposta do sistema com os polos realocados

Podemos perceber que as posições escolhidas para os polos mudaram a dinâmica do sistema fazendo com que o mesmo apresente um comportamento mais rápido e com um menor sobressinal.

O que resta agora é garantir que o erro em regime permanente seja o menor possível.

5.3. Coeficiente de erro $\boldsymbol{k_p}$

Agora, precisamos partir da diferença que existe entre a resposta do sistema apresentada na **Imagem 5.6** e do sinal de referência r, para este propósito foi utilizado método $control.matlab.dcgain(\,)$, o método em questão recebe a variável que contém nosso sistema com os polos realocados, sys_close, e retorna o valor de ganho do sistema.

Uma vez que o ganho do sistema é conhecido, podemos calcular a proporção em que precisamos multiplicar o valor na entrada para compensar o resultado do sistema na saída e atribuir esse valor multiplicativo ao módulo k_p do sistema. O trecho de código utilizado para este fim e as saídas obtidas podem ser observados abaixo:

```python
from control import matlab

Kdc = matlab.dcgain(sys_close) ##Ganho do sistema
Kp = 1/Kdc ##calculo do coeficiente de erro

print('VALOR DO COEFICIENTE DE ERRO: ')
print(Kp)
```

Imagem 5.7: Trecho de código utilizado para calcular k_p

```
VALOR DO COEFICIENTE DE ERRO:
50.49999999999999
```

Imagem 5.8: Valor calculado para k_p

5.4. Matriz B em malha fechada

Partindo do diagrama de blocos apresentado na **Imagem 3.1** e da eq 3.1 temos que em malha fechada a matriz B do sistema vai assumir o seguinte formato:

$$B_{cl} = Bk_p$$

Dessa forma, podemos partir da matriz B do sistema em malha aberta em conjunto de k_p, que foi calculado no passo anterior, para determinarmos B_{cl}. O trecho de código utilizado para este fim e a saída obtida podem ser observados abaixo:

```python
Bcl = np.dot(Kp,B) ## B = (B*kp)
print('MATRIZ B EM MALHA FECHADA: ')
print(Bcl)
```

Imagem 5.9: Cálculo da matriz B em malha fechada

```
MATRIZ B EM MALHA FECHADA:
[[ 0. ]
 [50.5]
 [ 0. ]]
```

Imagem 5.10: Matriz B do sistema em malha fechada

Portanto, partindo da **Imagem 5.10**, temos que a matriz B_{cl} do sistema em malha fechada é do seguinte formato:

$$B_{cl} = Bk_p = \begin{pmatrix} 0 \\ 50.5 \\ 0 \end{pmatrix}$$

5.5. Montagem do sistema em Malha Fechada

Uma vez que fechamos a malha do sistema do servo-controlador com os devidos valores para os módulos k e k_p no sistema, podemos garantir as matrizes A_{cl} e B_{cl}. Logo, se faz possível garantir a dinâmica esperada para o sistema além de garantir também como o mesmo vai responder na saída quando alimentado com um sinal de referência.

Partindo do método *control.matlab.ss*( ) e das matrizes A_{cl} e B_{cl} podemos montar a representação do nosso sistema em malha fechada dentro da variável *sys_close_kp*. O trecho de código utilizado pode ser observado abaixo:

```python
from control import matlab

sys_close_kp = matlab.ss(Acl,Bcl,C,D) ##Sistema em malha fechada
```

Imagem 5.11: Trecho de código utilizado para montar o sistema em malha fechada

6. Dinâmica do sistema em Malha Fechada

Uma vez que toda a dinâmica do servo-controlador está montada, incluindo a sequência lógica de execução e os valores desejados dentro dos blocos, podemos representa-la na variável *sys_close_kp* e avaliar a resposta ao sinal de referência, possibilitando observar que o sistema apresenta o comportamento desejado.

```python
from control import matlab

y, t, x = matlab.lsim(sys_close_kp , u, tempo) ##Resposta

##Resposta - Sistema em malha fechada
plt.plot(t, y,'black')
plt.grid(alpha = 0.5)
plt.title('SISTEMA EM MALHA FECHADA')
plt.ylabel('Amplitude')
plt.xlabel('Tempo(s)')
plt.show()
```

Imagem 5.12: Trecho de código utilizado para calcular e plotar a resposta do sistema em malha fechada

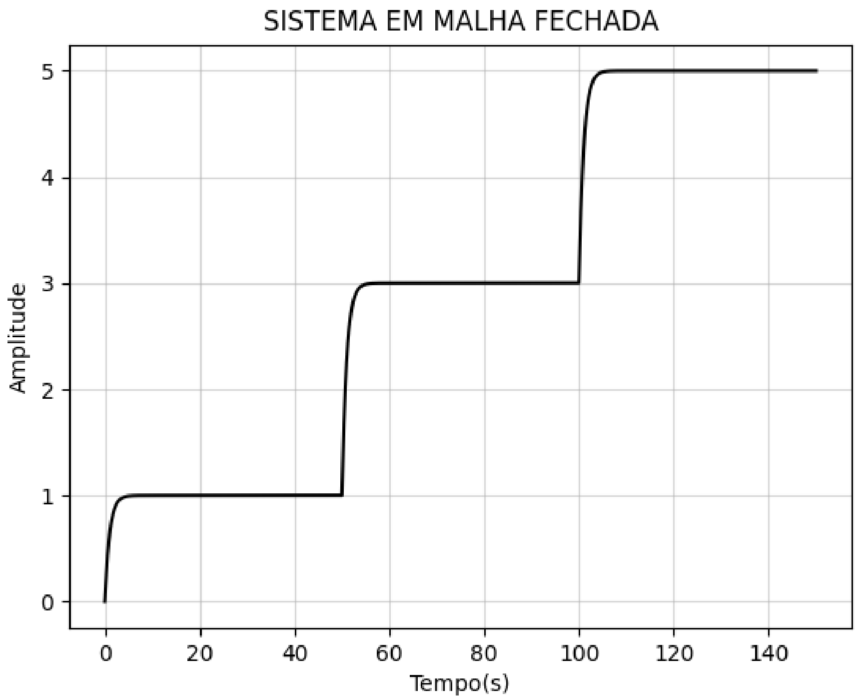

Imagem 5.13: Resposta em malha fechada com o sistema servo-controlador implementado

7. Critérios de desempenho (Malha Aberta x Malha Fechada)

Utilizando o método $control.matlab.step\_info()$ tanto no sistema em malha aberta(sys_open) quanto em malha fechada(sys_close_kp), podemos avaliar e comparar os critérios de desempenho. A tabela abaixo ilustra um comparativo entre o retorno do método para os dois casos:

	Malha Aberta	*Malha Fechada*
Overshoot	27.41	0
Valor de Pico	1.27	1
Tempo de Pico	$1.64s$	$6.9s$
Tempo de Subida	$1.13s$	$2.14s$
Tempo de Acomodação	$15.30s$	$3.18s$

Tabela 7.1: Comparativo entre os critérios de desempenho dos sistemas

Apesar do sistema em malha aberta possuir um tempo de subida e um tempo de pico menor o sistema em malha fechada possui um tempo de acomodação quase cinco vezes menor e um valor de pico que não ultrapassa o sinal de referência (degrau unitário) com o um ***overshoot*** de 0. Os valores da tabela estão ilustrados graficamente abaixo.

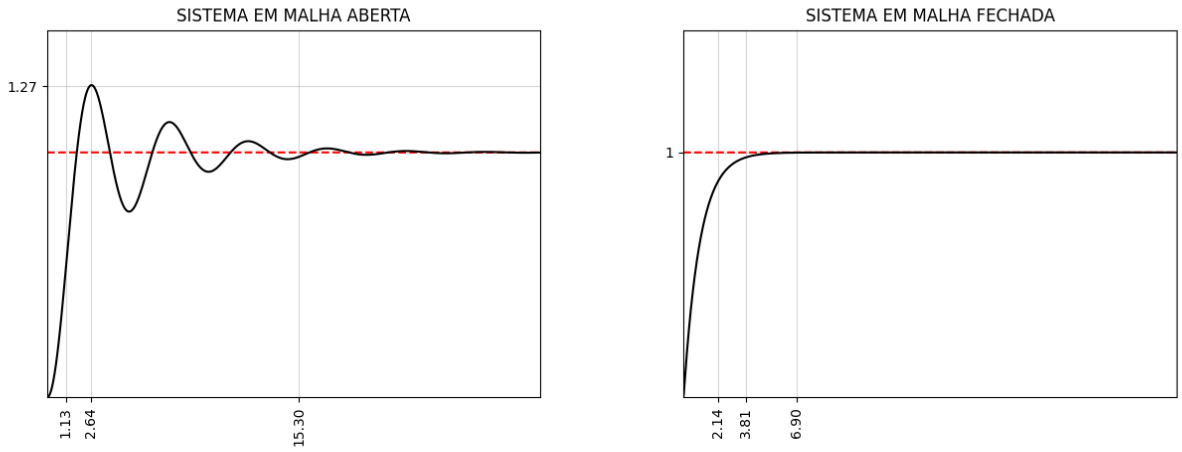

Imagem 7.1: Representação gráfica dos critérios de desempenho dos sistemas

8. Anexos

ANEXO A – Notebook com as implementações realizadas para a análise deste documento:

< Sistemas de Controle 2 - Parte 2 - AB1 - Colaboratory (google.com) >

9. Bibliografia

NISE, Norman. Engenharia de Sistemas de Controle. 6ª ED. Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora Ltda, 2012.

OGATA, Katsushiro. Engenharia de controle Moderno. 5ª ED. São Paulo: Pearson Education do Brasil, 2011.

KLUEVER, Craig. Sistemas Dinâmicos – Modelagem, Simulação e Controle. 1ª ED. Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora Ltda, 2017.

PYTHON SOFTWARE FOUNDATION. Python Language Site: Python Control Systems Library, 2022. Página de documentação. Disponível em: <https://python-control.readthedocs.io/en/0.9.1/>. Acesso em: 06 de Maio. de 2022.

PYTHON SOFTWARE FOUNDATION. Python Language Site: Signal processing, 2022. Página de documentação. Disponível em: < https://docs.scipy.org/doc/scipy/reference/signal.html>. Acesso em: 06 de Maio. de 2022.