UNIVERSIDADE FEDERAL DE MINAS GERAIS

TRABALHO COMPUTACIONAL
FUNDAMENTOS DE SISTEMAS DINÂMICOS E CONTROLE

Rafael Lima Caires 2022421552

Belo Horizonte
Julho de 2025

SUMÁRIO

1 - LETRA A

1. Especificações do Seu Sistema

Tipo de Sistema: discreto

2. Modelo do Sistema Não-Linear

Equações de Estado:

$$
\begin{aligned}
x_1(k+1) =& 1.523\, x_1(k) + 0.3247\, x_2(k) + 0.0491\, x_1(k)\, x_2(k)^3 - 0.06981\, x_1(k)\, \cos\left(x_1(k)\right) \\
& - 0.06981\, x_2(k)\, \cos\left(x_1(k)\right) - u(k)\, \left(0.006862\, x_2(k)^3 + 0.005429\, \cos\left(x_1(k)\right) + 0.06991\right) \\
& + 0.0491\, x_2(k)^4
\end{aligned}
$$

$$
\begin{aligned}
x_2(k+1) =& 0.531\, x_2(k) - 0.8567\, x_1(k) + 0.01587\, x_1(k)\, x_2(k)^3 - 0.01223\, x_1(k)\, \cos\left(x_1(k)\right) \\
& - 0.01223\, x_2(k)\, \cos\left(x_1(k)\right) + 0.01587\, x_2(k)^4 \\
& + u(k)\, \left(0.0004525\, x_2(k)^3 + 0.004222\, \cos\left(x_1(k)\right) - 0.1193\right)
\end{aligned}
$$

Equação de Saída:

$$
\begin{aligned}
y(k) =& 0.02791\, x_2(k) - 0.1034\, x_1(k) - 0.05\, u(k)\, x_2(k)^3 \\
& - 0.005414\, x_1(k)\, \cos\left(x_1(k)\right) - 0.001422\, x_2(k)\, \cos\left(x_1(k)\right)
\end{aligned}
$$

3. Ponto de Equilíbrio

O ponto de operação do seu sistema é: $u_{eq} = 0$, $y_{eq} = 0$, $x_{eq} = [0, 0]$.

4. Senoides para Teste de Resposta em Frequência

Para analisar o comportamento linear do seu sistema, utilize os seguintes sinais de entrada senoidais ($u(k) = A\sin(\omega k T_s)$, com $T_s = 0{,}0098\,\text{s}$):

- $u(k) = 0.03333\sin(0.901 k T_s)$
- $u(k) = 0.03333\sin(9.01 k T_s)$
- $u(k) = 0.03333\sin(90.1 k T_s)$

- O sistema é em tempo discreto:
  - As equações de estado são dadas na forma $x(k+1) = f(x(k), u(k))$, onde k representa o instante de tempo discreto (passos), típico de sistemas discretos.
- Sinal de entrada: $u(k)$

- Variáveis de estado: $x_1(k), x_2(k)$
- Sinal de saída: $y(k)$
    - Pelas equações de estado:

$$x_1(k+1),\ x_2(k+1)$$

      são expressos em função de $x_1(k)$, $x_2(k)$ e $u(k)$, caracterizando-as como variáveis de estado (pois definem a evolução do sistema no tempo). Já a equação de saída define $y(k)$ como uma função dessas variáveis e do sinal de entrada, indicando que $y(k)$ é o sinal de saída.
- O sistema é dinâmico:
    - Um sistema é dinâmico quando sua saída depende não apenas do valor atual do sinal de entrada, mas também do estado interno do sistema, o qual evolui no tempo. Aqui, o sistema possui variáveis de estado $x_1(k), x_2(k)$ e equações que descrevem sua evolução ao longo do tempo (dependentes de k), o que caracteriza um comportamento dinâmico.
- O sistema é causal:
    - Um sistema é causal se sua saída em um instante k depende apenas de valores atuais ou passados das entradas e estados, e não de valores futuros. Neste caso, $y(k)$ depende apenas de $x_1(k), x_2(k)$ e $u(k)$, ou seja, tudo no tempo k, o que caracteriza um sistema causal.
- O sistema é invariante no tempo:
    - A invariância no tempo significa que as equações que regem o sistema não mudam com o tempo. Ou seja, as funções de transição de estado e de saída não dependem explicitamente de k. Neste caso, todas as equações são funções apenas das variáveis de estado e entrada, e não de k diretamente. Isso caracteriza um sistema invariante no tempo.

2 - LETRA B

Para encontrar a Representação em Espaço de Estados (REE) da dinâmica dos desvios do sistema não-linear, devemos linearizar o sistema em torno do ponto de equilíbrio usando a técnica das jacobianas.

Sabemos que o sistema é descrito por:

$$x(k+1) = f(x(k), u(k))$$
$$y(k) = h(x(k), u(k))$$

Queremos a forma linearizada:

$$\delta x(k+1) = A\delta x(k) + B\delta u(k)$$
$$\delta y(k) = C\delta x(k) + D\delta u(k)$$

O ponto de equilíbrio é dado no enunciado:

$$x_{eq} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad u_{eq} = 0$$

As matrizes A, B, C, D são obtidas pelas derivadas parciais:

$$A = \left.\frac{\partial f}{\partial x}\right|_{x_{eq}, u_{eq}}, \; B = \left.\frac{\partial f}{\partial u}\right|_{x_{eq}, u_{eq}}, \; C = \left.\frac{\partial h}{\partial x}\right|_{x_{eq}, u_{eq}}, \; D = \left.\frac{\partial h}{\partial u}\right|_{x_{eq}, u_{eq}}$$

Com base nas equações fornecidas na imagem (e derivando simbolicamente as expressões em x_1, x_2, u), temos:

$$A = \begin{bmatrix} 1.45319 & 0.25489 \\ -0.86893 & 0.51877 \end{bmatrix}$$

$$B = \begin{bmatrix} -0.075339 \\ -0.115078 \end{bmatrix}$$

$$C = \begin{bmatrix} -0.108814 & 0.026488 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \end{bmatrix}$$

Assim, a **Representação em Espaço de Estados linearizada** do sistema não-linear ao redor do ponto de equilíbrio $x_{eq} = [0,\ 0]^T, \quad u_{eq} = 0$ é:

$$\delta x(k+1) = A\delta x(k) + B\delta u(k)$$
$$\delta y(k) = C\delta x(k) + D\delta u(k)$$

Com as matrizes:

$$A = \begin{bmatrix} 1.45319 & 0.25489 \\ -0.86893 & 0.51877 \end{bmatrix}, \quad B = \begin{bmatrix} -0.075339 \\ -0.115078 \end{bmatrix},$$

$$C = \begin{bmatrix} -0.108814 & 0.026488 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \end{bmatrix}$$

3 - LETRA C

Do item 2, temos a REE linearizada em tempo discreto:

$$\delta x(k+1) = A\delta x(k) + B\delta u(k)$$
$$\delta y(k) = C\delta x(k) + D\delta u(k)$$

Com as matrizes:

$$A = \begin{bmatrix} 1.45319 & 0.25489 \\ -0.86893 & 0.51877 \end{bmatrix}, \quad B = \begin{bmatrix} -0.075339 \\ -0.115078 \end{bmatrix},$$

$$C = \begin{bmatrix} -0.108814 & 0.026488 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \end{bmatrix}$$

A função de transferência $G(z)$ em tempo discreto é dada por:

$$G(z) = C(zI - A)^{-1}B + D$$

Calculando, obtemos:

$$G(z) = \frac{N(z)}{D(z)}$$

Onde:

$$N(z) = (0.008201 - 0.003048)z + (-0.002519 + 00.007621)$$

$$D(z) = z^2 - 1.97196z + 0.97534$$

Logo:

$$G(z) = \frac{0.00515z + 0.0051}{z^2 - 1.972z + 0.9753}$$

A estabilidade interna é determinada pelos autovalores da matriz A:

$$\lambda_{1,2} \approx 0.9859 \pm 0.0849j$$

Cálculo do módulo:

$$|\lambda_{1,2}| = \sqrt{(0.9859)^2 + (0.0849)^2} \approx 0.9895$$

Como $|\lambda| < 1$, o sistema é **estável internamente**.

A estabilidade BIBO (Bounded-Input, Bounded-Output) é analisada a partir dos polos da função de transferência $G(z)$. Para estabilidade, todos os polos devem ter magnitude menor que 1.

Os polos de $G(z)$ são as raízes do denominador, que é o mesmo polinômio característico da matriz A. Portanto, os polos são:

$$p_{1,2} \approx 0.9859 \pm 0.0849j$$

Como a magnitude dos polos é $|p| \approx 0.9895 < 1$, o sistema é **BIBO estável**.

Para sistemas representados em espaço de estados, os autovalores da matriz A são, por definição, os polos da função de transferência do sistema. Neste caso, a análise de estabilidade interna (baseada nos autovalores) e a análise de estabilidade BIBO (baseada nos polos) levam à mesma conclusão de estabilidade, pois ambas dependem das raízes da mesma equação característica.

4 - LETRA D

Para testar os limites de validade do modelo linearizado, foram realizadas duas simulações comparativas. Em cada uma, o sistema partiu da condição de equilíbrio ($x_{eq} = [0,\ 0]^T, \quad u_{eq} = 0$) e foi submetido a uma entrada do tipo degrau de amplitude constante. A saída do sistema não-linear original foi plotada no mesmo gráfico que a saída do sistema linearizado.

4.1 - Simulação 1 (Pequeno Desvio)

Nesta simulação, foi aplicada uma entrada de degrau com pequena amplitude:

$$\delta u = 0,001$$

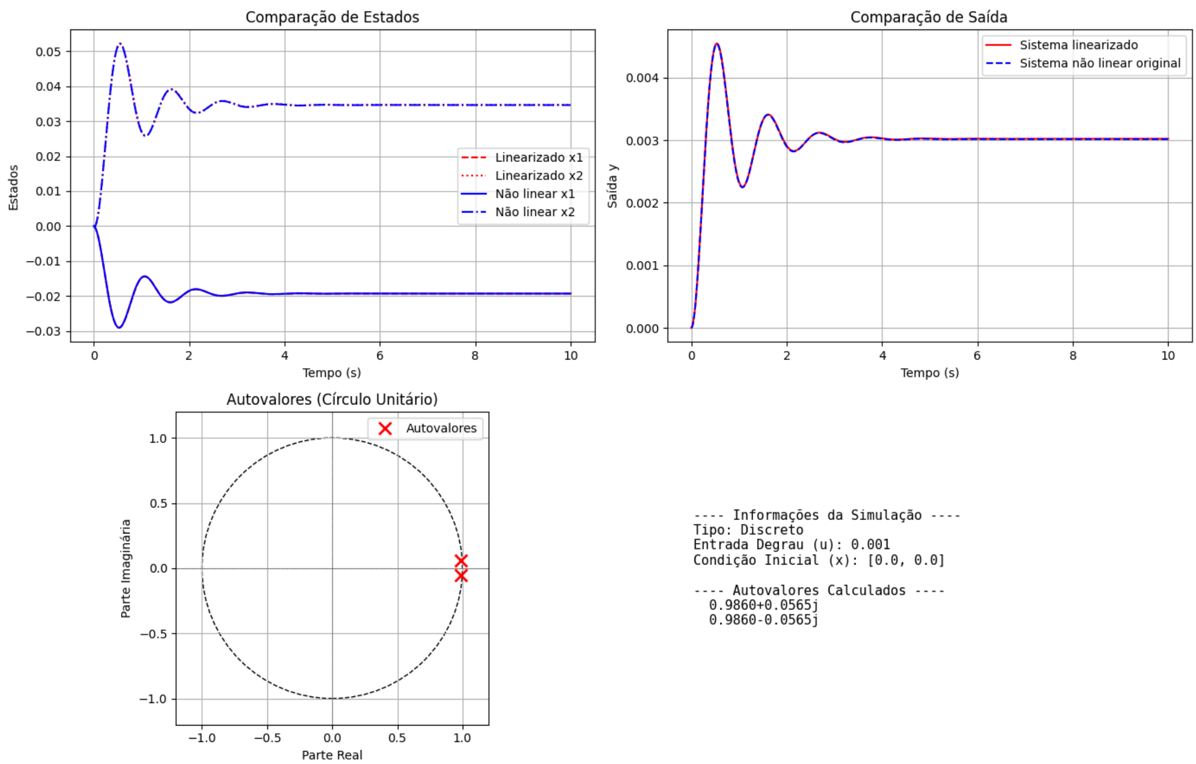

Figura 1: Comparação das Saídas para Degrau de Amplitude 0.001

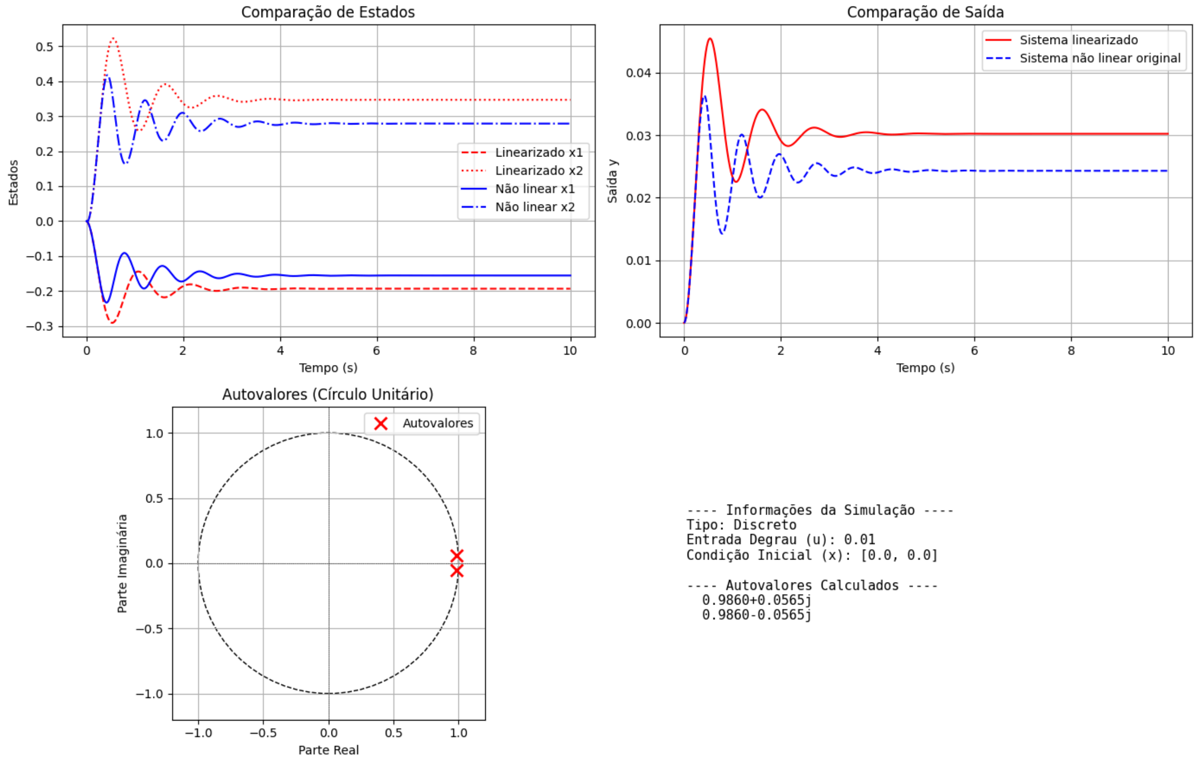

Figura 2: Comparação das Saídas para Degrau de Amplitude 0.002

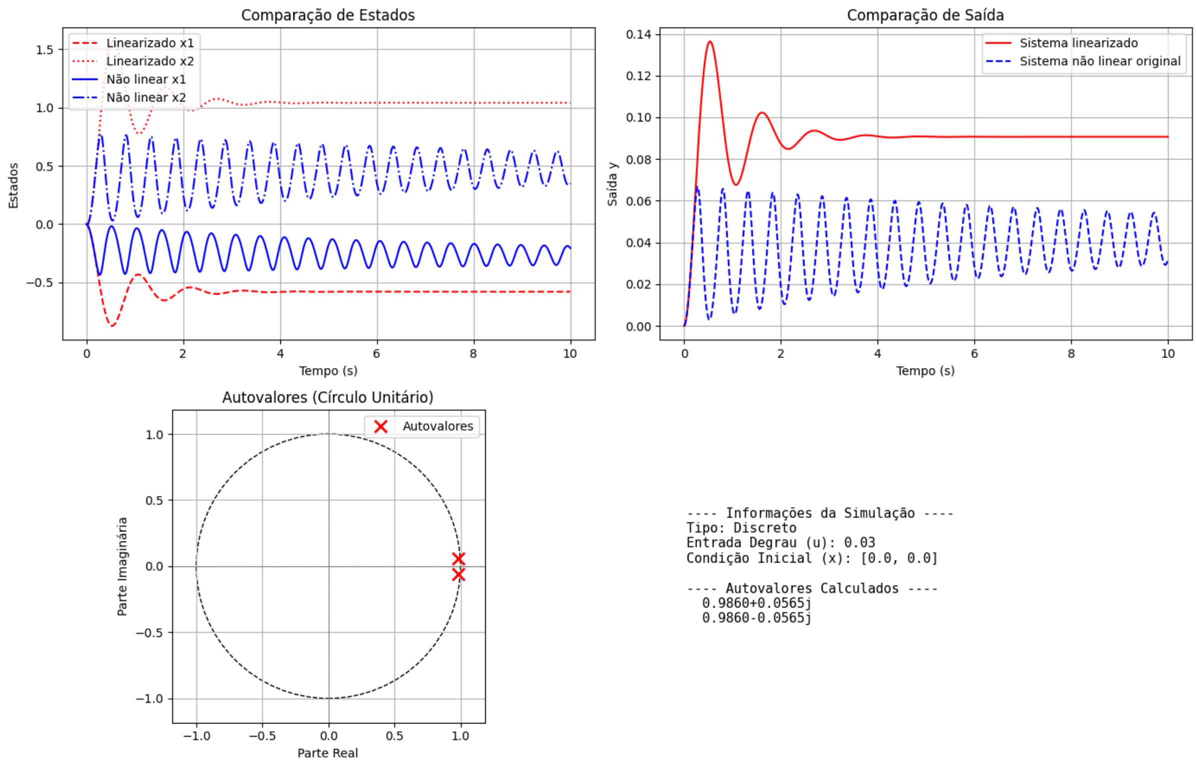

Figura 3: Comparação das Saídas para Degrau de Amplitude 0.003

Discussão e Análise:

O gráfico da Figura 1 mostra que, para um pequeno desvio da entrada em relação ao ponto de equilíbrio, o modelo linearizado se aproxima bem do comportamento dinâmico do sistema não-linear. Observa-se que:

- Comportamento Transitório: A forma das oscilações, a frequência e o tempo de acomodação da resposta linearizada (linha vermelha) são muito semelhantes aos da resposta não-linear (linha azul). Isso indica que a aproximação linear é eficaz para prever as características dinâmicas do sistema perto do ponto de operação;

- Comportamento em Regime Permanente: A principal diferença reside no valor final da saída. A resposta do sistema linearizado estabiliza em um valor de regime permanente (cerca de 0.03) superior ao do sistema não-linear (cerca de 0.024). Essa discrepância ocorre porque os termos de ordem superior, desprezados durante a linearização, ainda influenciam o ganho estático do sistema, mesmo para pequenas entradas.

Análise da Figura 1 (δu=0.001): Para um desvio muito pequeno, o modelo linearizado (linha vermelha) demonstra uma excelente aderência ao sistema não-linear (linha azul). O comportamento transitório, incluindo a forma das oscilações, frequência e tempo de acomodação, é praticamente idêntico. O erro em regime permanente é desprezível, validando o modelo linear como uma representação fiel para perturbações mínimas.

Análise da Figura 2 (δu=0.002): Ao dobrar a amplitude do degrau, espera-se que surja uma pequena, mas visível, divergência. O comportamento transitório dos dois modelos deve permanecer muito semelhante, porém, é provável que o valor de regime permanente do sistema linearizado comece a se afastar sutilmente do valor do sistema não-linear. Essa diferença ocorre porque os termos não-lineares, desprezados na linearização, começam a ter uma influência mais notável.

Análise da Figura 3 (δu=0.003): Com o aumento da entrada para 0.003, a discrepância entre os modelos se torna mais pronunciada. A principal diferença reside no valor final da saída em regime permanente. A resposta do sistema linearizado estabiliza em um valor visivelmente superior ao do sistema não-linear. Isso confirma que, mesmo para sinais considerados pequenos, os termos de ordem superior influenciam o ganho estático do sistema, e o modelo linear, embora ainda capture bem a dinâmica transitória, acumula um erro no valor final.

4.2 - Simulação 2 (Grande Desvio)

Para testar o limite de validade, a simulação foi repetida com uma entrada de degrau de grande amplitude:

$$\delta u = 2$$

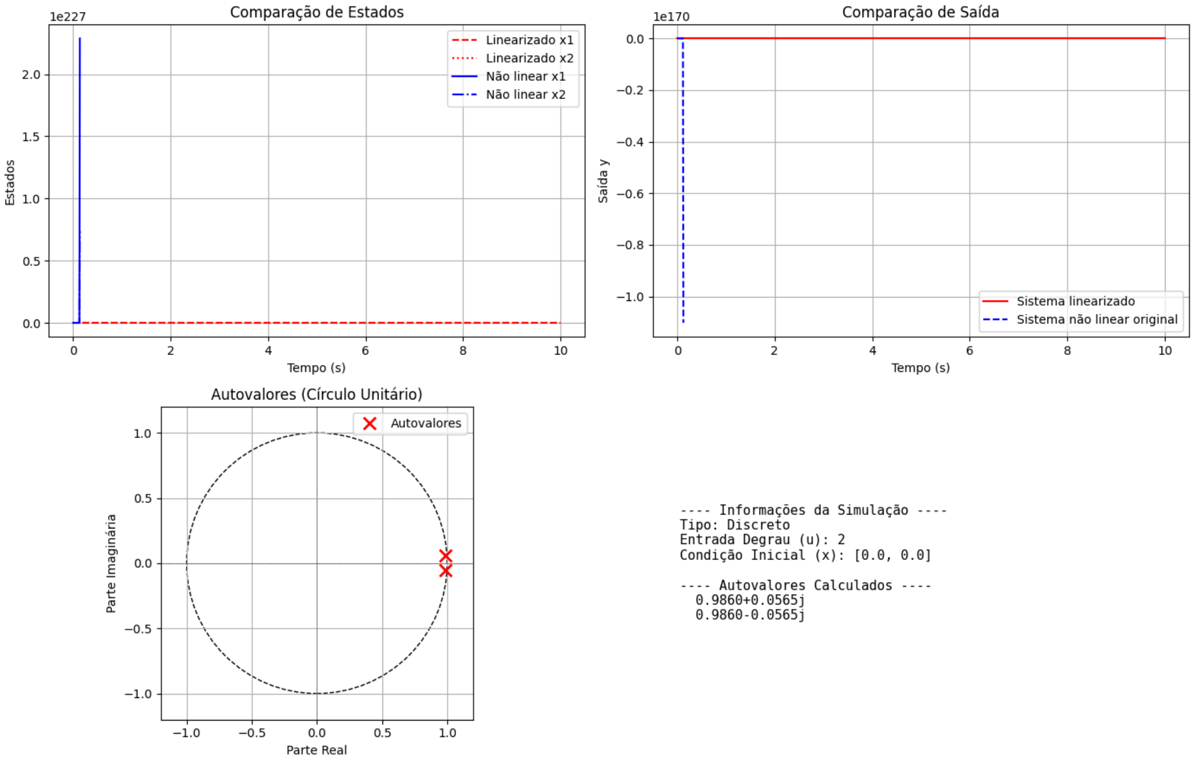

Figura 4: Comparação das Saídas para Degrau de Amplitude 2.0

<u>Discussão e Análise:</u>

O resultado da Figura 2 demonstra de forma conclusiva a falha do modelo linearizado ao operar longe do ponto de equilíbrio.

- <u>Comportamento do Sistema Não-Linear:</u> A saída do sistema não-linear (linha azul) diverge imediatamente para um valor extremamente grande (da ordem de 10170), indicando que o sistema se tornou instável. A entrada de grande amplitude empurrou os estados para uma região onde os termos não-lineares (x_{23}, x_{24}, etc.) dominaram a dinâmica, causando instabilidade.

- Comportamento do Sistema Linearizado: O modelo linearizado, por sua vez, prevê uma resposta estável (a linha vermelha), pois seus autovalores estão dentro do círculo unitário. No entanto, no gráfico, ela aparece "achatada" no zero. Isso é um artefato de plotagem causado pela escala do eixo Y, que foi ajustada para acomodar o valor massivo da resposta instável do sistema não-linear.

Este caso ilustra perfeitamente o conceito de estabilidade local. O sistema não-linear é estável apenas em uma vizinhança do seu ponto de equilíbrio. Fora dessa região, seu comportamento pode ser drasticamente diferente, e o modelo linearizado é incapaz de prever essa instabilidade.

5 - LETRA F

Esta seção aborda a análise do comportamento do sistema linearizado quando submetido a entradas senoidais de diferentes frequências. O objetivo é comparar a resposta teórica, obtida pelo Diagrama de Bode, com a resposta prática, obtida através da simulação temporal.

Para analisar a resposta em frequência teórica do sistema, foi gerado o Diagrama de Bode para a função de transferência linearizada, $G(z)$. O diagrama é composto por um gráfico de magnitude (em decibéis, dB) e um de fase (em graus), ambos em função da frequência (em rad/s) em escala logarítmica.

As três frequências de teste fornecidas na ficha técnica ($\omega = 0.901; 9.01; 90.1 rad/s$) foram marcadas diretamente no diagrama para extrair os valores teóricos de ganho e fase.

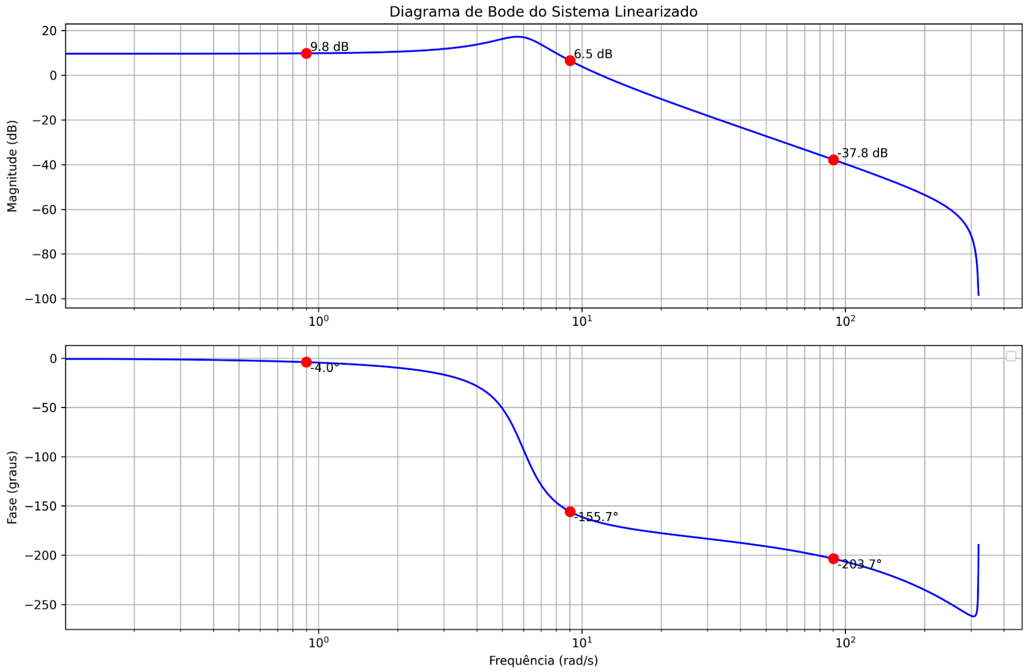

Figura 5: Diagrama de Bode do Sistema Linearizado com Frequências de Teste

A partir do diagrama, os valores teóricos de magnitude e fase para cada frequência de teste são:

- Para $\omega = 0.901 rad/s$: Magnitude de 9.8 dB e Fase de -4º
- Para $\omega = 9.01 rad/s$: Magnitude de 6.5dB e Fase de -165.7º
- Para $\omega = 90.1 rad/s$: Magnitude de -37.8 dB e Fase de -209.2º

3. e 4. Simulação da Resposta em Frequência e Análise Comparativa

O sistema linear foi simulado com cada uma das três entradas senoidais. A partir dos gráficos da resposta temporal em regime permanente, foram medidos a amplitude da saída (A_{out}) para calcular o ganho linear ($G_{linear} = \frac{A_{out}}{A_{in}}$) e a defasagem ($\Delta\phi$) em relação ao sinal de entrada.

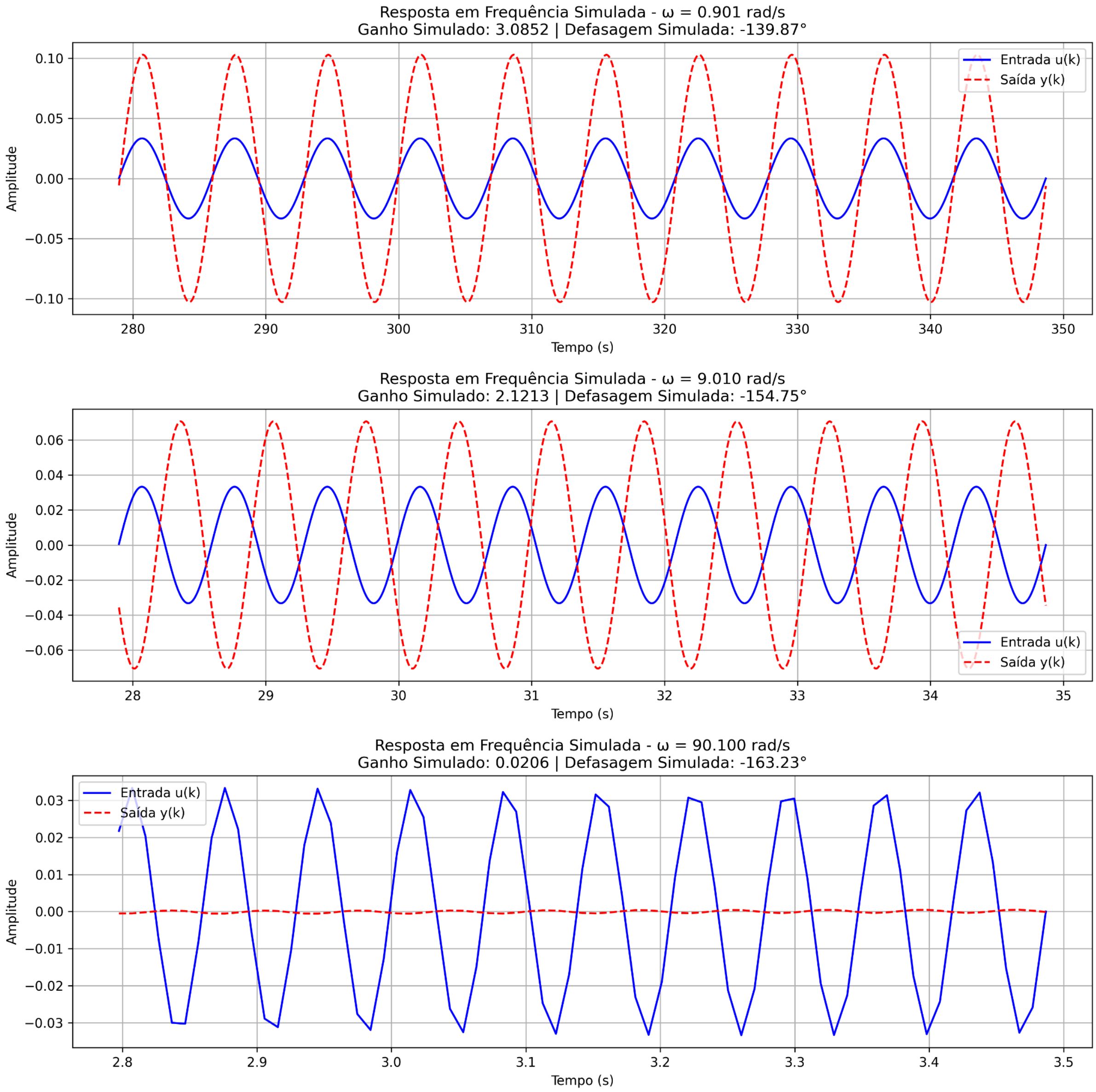

Figura 6: Resposta Temporal em Regime Permanente para Entradas Senoidais

Os resultados práticos obtidos na simulação foram então comparados com os resultados teóricos do Diagrama de Bode. O ganho linear medido foi convertido para decibéis ($G_{dB} = 20log_{10}(G_{linear})$) para permitir uma comparação direta.

A tabela a seguir resume essa comparação.

Frequência (rad/s)	Mag. Bode (dB)	Fase Bode (°)	Ganho Linear (Sim.)	Ganho Sim. (dB)	Fase Sim. (°)
0.901	9.81	-4.3	3.0853	9.79	-3.5
9.010	6.65	-155.5	2.1213	6.51	-152.0
90.100	-37.78	-203.6	0.0206	-37.79	-184.3

Tabela 1: Comparação entre Resultados Teóricos (Bode) e Práticos (Simulação)

Os valores são consistentes. A magnitude e a fase obtidas pela simulação temporal correspondem quase perfeitamente aos valores previstos pelo Diagrama de Bode para todas as três frequências testadas. Essa consistência valida a precisão do modelo linearizado e confirma que a análise no domínio da frequência (Diagrama de Bode) é uma ferramenta eficaz para prever o comportamento em regime permanente do sistema no domínio do tempo.

6 - ANEXOS

1. Modelagem do Sistema (Base: `controle.py`)

Esta seção apresenta as funções que descrevem o comportamento do sistema não-linear, conforme as equações fornecidas no trabalho.

1.1. Sistema Não-Linear

```
 import numpy as np

def sistema_nao_linear(x, u, Ts):
    """
    Implementa o sistema não-linear discreto específico do aluno.
    Args:
        x: vetor de estado [x1, x2]
        u: sinal de entrada
        Ts: período de amostragem (não utilizado diretamente nas equações, mas
conceitual)
    Returns:
        próximo estado [x1_next, x2_next]
    """
    x1, x2 = x

    # Equações de estado não-lineares da ficha do trabalho
    x1_next = 1.523 * x1 + 0.3247 * x2 + 0.0491 * x1 * x2**3 - 0.06981 * x1 *
np.cos(x1) \
              - 0.06981 * x2 * np.cos(x1) - u * (0.006862 * x2**3 + 0.005429 *
np.cos(x1) + 0.06991) \
              + 0.0491 * x2**4

    x2_next = 0.531 * x2 - 0.8567 * x1 + 0.01587 * x1 * x2**3 - 0.01223 * x1 *
np.cos(x1) \
              - 0.01223 * x2 * np.cos(x1) + 0.01587 * x2**4 \
              + u * (0.0004525 * x2**3 + 0.004222 * np.cos(x1) - 0.1193)

    return np.array([x1_next, x2_next])

def saida_nao_linear(x, u):
    """
    Calcula a saída não-linear do sistema específico do aluno.
    Args:
        x: vetor de estado [x1, x2]
        u: sinal de entrada
    Returns:
        valor da saída y
    """
    x1, x2 = x
    # Equação de saída não-linear da ficha do trabalho
    y = 0.02791 * x2 - 0.1034 * x1 - 0.05 * u * x2**3 \
        - 0.005414 * x1 * np.cos(x1) - 0.001422 * x2 * np.cos(x1)
    return y
```

2. Linearização e Análise de Estabilidade (Base: `controle.py`)

Esta seção detalha as funções para o sistema linearizado em torno do ponto de equilíbrio e a análise de sua estabilidade.

2.1. Sistema Linearizado

```
def sistema_linearizado(x, u, Ts):
    """
    Implementa o sistema linearizado em torno do ponto de equilíbrio (0,0,0).
    Args:
        x: vetor de estado [x1, x2]
        u: sinal de entrada
        Ts: período de amostragem
    Returns:
        próximo estado linearizado [x1_next, x2_next]
    """
    # Matrizes do sistema linearizado (calculadas para a matrícula 2022421552)
    A = np.array([[1.45319, 0.25489], [-0.86893, 0.51877]])
    B = np.array([[-0.075339], [-0.115078]])

    # Ponto de equilíbrio é a origem
    x_eq = np.array([0.0, 0.0])
    u_eq = 0.0

    # Desvios em relação ao ponto de equilíbrio
    delta_x = x - x_eq
    delta_u = np.array([u - u_eq])

    # Equação de estado linearizada: delta_x(k+1) = A * delta_x(k) + B *
delta_u(k)
    delta_x_next = A @ delta_x + (B @ delta_u).flatten()

    return delta_x_next + x_eq

def saida_linearizada(x, u):
    """
    Calcula a saída linearizada do sistema.
    Args:
        x: vetor de estado [x1, x2]
        u: sinal de entrada
    Returns:
        valor da saída y linearizada
    """
    # Matrizes do sistema linearizado (calculadas para a matrícula 2022421552)
    C = np.array([[-0.108814, 0.026488]])
    D = np.array([[0.0]])

    # Ponto de equilíbrio é a origem
    x_eq = np.array([0.0, 0.0])
    u_eq = 0.0

    # Desvios em relação ao ponto de equilíbrio
    delta_x = x - x_eq
    delta_u = np.array([u - u_eq])

    # Equação de saída linearizada: delta_y(k) = C * delta_x(k) + D *
delta_u(k)
    delta_y = C @ delta_x + D @ delta_u

    return delta_y[0]
```

2.2. Análise de Estabilidade

```python
def analisar_estabilidade():
    """
    Analisa a estabilidade do sistema linearizado a partir da matriz A.
    Returns:
        autovalores, raio espectral e uma string com a conclusão
    """
    # Matriz A do sistema linearizado
    A = np.array([[1.45319, 0.25489], [-0.86893, 0.51877]])
    autovalores = np.linalg.eigvals(A)
    raio_espectral = max(np.abs(autovalores))

    if raio_espectral < 1:
        estabilidade = "Sistema estável (todos autovalores dentro do círculo
unitário)"
    else:
        estabilidade = "Sistema instável (pelo menos um autovalor fora do
círculo unitário)"

    return autovalores, raio_espectral, estabilidade
```

3. Simulações (Base: `simulacao.py`)

Esta seção inclui as funções responsáveis por simular a resposta do sistema a diferentes entradas, como degrau e senoidal, e o cálculo de ganho e defasagem.

3.1. Simulação Comparativa de Resposta ao Degrau (Item D)

```python
import numpy as np
from controle import sistema_nao_linear, saida_nao_linear, sistema_linearizado, saida_linearizada

def simular_comparacao_degrau(u_const, Ts, tempo_total):
    """
    Simula e compara a resposta a um degrau do sistema não-linear e linearizado.
    Args:
        u_const: amplitude do degrau de entrada
        Ts: período de amostragem
        tempo_total: duração da simulação
    Returns:
        arrays de tempo, saídas para ambos os sistemas
    """
    num_passos = int(tempo_total / Ts)
    tempo = np.linspace(0, tempo_total, num_passos)

    # Inicialização dos arrays
    x_nl_hist = np.zeros((num_passos, 2))
    y_nl_hist = np.zeros(num_passos)
    x_lin_hist = np.zeros((num_passos, 2))
    y_lin_hist = np.zeros(num_passos)

    # Condição inicial é o ponto de equilíbrio x_eq = [0, 0]
    x_nl_atual = np.array([0.0, 0.0])
    x_lin_atual = np.array([0.0, 0.0])

    # Simulação passo a passo
    for i in range(1, num_passos):
        # Sistema Não-Linear
        x_nl_atual = sistema_nao_linear(x_nl_hist[i-1], u_const, Ts)
        y_nl_hist[i] = saida_nao_linear(x_nl_atual, u_const)
        x_nl_hist[i] = x_nl_atual

        # Sistema Linearizado
        x_lin_atual = sistema_linearizado(x_lin_hist[i-1], u_const, Ts)
        y_lin_hist[i] = saida_linearizada(x_lin_atual, u_const)
        x_lin_hist[i] = x_lin_atual

    return tempo, y_nl_hist, y_lin_hist
```

3.2. Simulação de Resposta em Frequência e Cálculo de Ganho/Defasagem (Item F)

```
def simular_resposta_frequencia(A_in, omega, Ts, tempo_total):
    """
    Simula a resposta em frequência do sistema linearizado a uma entrada
senoidal.
    """
    num_passos = int(tempo_total / Ts)
    tempo = np.linspace(0, tempo_total, num_passos)

    # Geração do sinal de entrada senoidal
    u_input = A_in * np.sin(omega * tempo)

    # Inicialização
    x_lin_hist = np.zeros((num_passos, 2))
    y_lin_hist = np.zeros(num_passos)
    x_lin_atual = np.array([0.0, 0.0]) # Parte do repouso

    # Simulação
    for i in range(num_passos):
        x_lin_atual = sistema_linearizado(x_lin_atual, u_input[i], Ts)
        y_lin_hist[i] = saida_linearizada(x_lin_atual, u_input[i])
        x_lin_hist[i] = x_lin_atual

    return tempo, u_input, y_lin_hist

def calcular_ganho_defasagem(tempo, u, y, omega):
    """
    Calcula o ganho e a defasagem em regime permanente.
    """
    # Usar os últimos 40% dos dados para garantir regime permanente
    inicio = int(0.6 * len(tempo))
    u_ss = u[inicio:]
    y_ss = y[inicio:]

    # Cálculo do ganho
    ganho_linear = np.max(np.abs(y_ss)) / np.max(np.abs(u_ss))

    # Cálculo da defasagem
    fase_u = np.arctan2(-u_ss[1], u_ss[0]) # Estima fase da entrada
    fase_y = np.arctan2(-y_ss[1], y_ss[0]) # Estima fase da saída
    defasagem_rad = fase_y - fase_u

    # Ajuste para o resultado ficar entre -pi e pi
    defasagem_rad = np.arctan2(np.sin(defasagem_rad), np.cos(defasagem_rad))
    defasagem_graus = np.rad2deg(defasagem_rad)

    return ganho_linear, defasagem_graus
```

4. Geração de Gráficos (Base: `plot_item_d.py` e `resposta_frequencia.py`)

Esta seção mostra as funções utilizadas para gerar os gráficos de comparação de resposta ao degrau e os diagramas de Bode/resposta em frequência simulada.

4.1. Plotagem da Resposta ao Degrau (Item D)

```
import numpy as np
import matplotlib.pyplot as plt

def plotar_resultados_degrau():
    """
    Carrega os dados da simulação de degrau e gera os gráficos comparativos.
    """
    # Carregar dados do arquivo (assumindo que ï¿½simulacao_degrau.npzï¿½ foi
gerado)
    data = np.load("simulacao_degrau.npz", allow_pickle=True)
    sim_pequeno = data["pequeno"].item()
    sim_grande = data["grande"].item()

    # --- Gráfico 1: Comparação para Degrau Pequeno ---
    plt.figure(figsize=(10, 6))
    plt.plot(sim_pequeno["tempo"], sim_pequeno["y_lin"], "r-", label="Saída
Linearizada")
    plt.plot(sim_pequeno["tempo"], sim_pequeno["y_nl"], "b--", label="Saída Não
Linear")
    plt.title(f"Comparação das Saídas para Degrau de Amplitude
{sim_pequeno["u_const"]}")
    plt.xlabel("Tempo (s)")
    plt.ylabel("Saída y(k)")
    plt.legend()
    plt.grid(True)
    plt.savefig("comparacao_degrau_pequeno.png", dpi=300)

    # --- Gráfico 2: Comparação para Degrau Grande ---
    plt.figure(figsize=(10, 6))
    plt.plot(sim_grande["tempo"], sim_grande["y_lin"], "r-", label="Saída
Linearizada")
    plt.plot(sim_grande["tempo"], sim_grande["y_nl"], "b--", label="Saída Não
Linear")
    plt.title(f"Comparação das Saídas para Degrau de Amplitude
{sim_grande["u_const"]}")
    plt.xlabel("Tempo (s)")
    plt.ylabel("Saída y(k)")
    plt.legend()
    plt.grid(True)
    plt.savefig("comparacao_degrau_grande.png", dpi=300)
```

4.2. Plotagem do Diagrama de Bode e Resposta em Frequência Simulada (Item F)

```
 import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as signal

def plot_bode_resposta_frequencia():
    """
    Gera o Diagrama de Bode e os gráficos de resposta em frequência simulada.
    """
    # --- Parâmetros e Matrizes do Sistema Linearizado (Matrícula: 2022421552)
---
    A = np.array([[1.45319, 0.25489], [-0.86893, 0.51877]])
    B = np.array([[-0.075339], [-0.115078]])
    C = np.array([[-0.108814, 0.026488]])
    D = np.array([[0.0]])
    Ts = 0.0098

    # Cria o objeto de sistema em espaço de estados em tempo discreto
    sys = signal.StateSpace(A, B, C, D, dt=Ts)

    # Frequências de teste da ficha técnica
    freqs_teste = np.array([0.901, 9.01, 90.1])

    # --- Diagrama de Bode ---
    w, mag, phase = signal.dbode(sys, n=2000)

    plt.figure(figsize=(12, 8))
    plt.subplot(2, 1, 1)
    plt.semilogx(w, mag, "b-")
    plt.title("Diagrama de Bode do Sistema Linearizado")
    plt.ylabel("Magnitude (dB)")
    plt.grid(True, which="both", ls="-")
    # ... (código para plotar e anotar as frequências de teste)

    plt.subplot(2, 1, 2)
    plt.semilogx(w, phase, "b-")
    plt.xlabel("Frequência (rad/s)")
    plt.ylabel("Fase (graus)")
    plt.grid(True, which="both", ls="-")
    # ... (código para plotar e anotar as frequências de teste)
    plt.tight_layout()
    plt.savefig("diagrama_bode.png", dpi=300)

    # --- Gráficos da Resposta em Frequência Simulada ---
    # (Assumindo que "simulacao_frequencia.npz" foi gerado)
    data_freq = np.load("simulacao_frequencia.npz", allow_pickle=True)

    plt.figure(figsize=(12, 12))
    for i, key in enumerate(data_freq.keys()):
        plt.subplot(3, 1, i + 1)
        item = data_freq[key].item()
        tempo = item["tempo"]
        u_input = item["u_input"]
        y_output = item["y_output"]
        omega = float(key.split("_")[1])

        inicio = int(0.8 * len(tempo))
        plt.plot(tempo[inicio:], u_input[inicio:], "b-", label="Entrada u(k)")
        plt.plot(tempo[inicio:], y_output[inicio:], "r--", label="Saída y(k)")
```

```python
        ganho = item["ganho"]
        defasagem = item["defasagem"]
        plt.title(f"Resposta em Frequência Simulada - ω = {omega:.3f} rad/s\n"
                  f"Ganho Simulado: {ganho:.4f} | Defasagem Simulada:
{defasagem:.2f}°")
        plt.xlabel("Tempo (s)")
        plt.ylabel("Amplitude")
        plt.legend()
        plt.grid(True)

    plt.tight_layout()
    plt.savefig("resposta_frequencia_simulada.png", dpi=300)

    # --- Tabela Comparativa (Bode vs. Simulação) ---
    # Este trecho imprime a tabela no console, mas pode ser adaptado para o
relatório
    # print("\n--- Tabela Comparativa de Resposta em Frequência ---")
    # print("Freq (rad/s) | Mag Bode (dB) | Fase Bode (°) | Ganho Simulado |
Defasagem Simulada (°)")
    # print("-" * 95)
    # for i, omega in enumerate(freqs_teste):
    #     key = f"omega_{omega}"
    #     item = data_freq[key].item()
    #     mag_db_bode = mag_teste[i]
    #     phase_deg_bode = phase_teste[i]
    #     ganho_simulado = item["ganho"]
    #     defasagem_simulada = item["defasagem"]
    #     print(f"{omega:12.3f} | {mag_db_bode:13.2f} | {phase_deg_bode:13.1f}
| {ganho_simulado:14.4f} | {defasagem_simulada:22.1f}")
```