Relatório Técnico Detalhado

Projeto de Controle para Servomecanismo de Posição

Gabriel, Felipe, Cintia, Dierson, Guilherme, Nicolas

11 de dezembro de 2025

Conteúdo

1 Introdução **3**

2 Modelagem Matemática - Gabriel **3**
2.1 Função de Transferência . . . 3
2.2 Parâmetros do Sistema . . . 3
2.3 Análise de Malha Aberta . . . 3

3 Controlador Proporcional (P) - Felipe **4**
3.1 Especificações e Sintonia . . . 4
3.2 Desempenho (Simulação) . . . 5
3.2.1 Análise de Desempenho . . . 5

4 Compensador Lag (Atraso de Fase) - Dierson **6**
4.1 Cálculo do Erro em Rampa (Sem Compensação) . . . 6
4.2 Projeto do Compensador . . . 7
4.2.1 Análise de Ganho e Resultado (Dierson) . . . 7
4.3 Novo Cálculo de Erro . . . 7
4.4 Análise Frequencial e Temporal . . . 8

5 Compensador Lead (Avanço de Fase) - Cintia **10**
5.1 Projeto do Compensador . . . 10
5.2 Análise no Lugar das Raízes e Bode . . . 11
5.3 Desempenho Temporal . . . 13

6 Controlador Lead-Lag Integrado (Solução Combinada) - Conjunto **14**
6.1 Estratégia de Projeto Detalhada . . . 14
6.1.1 1. O "Acelerador" (Compensador Lead) . . . 14
6.1.2 2. O "Corretor" (Compensador Lag) . . . 14
6.1.3 3. Sintonia Fina do Ganho (K) . . . 14
6.2 Resultados Finais . . . 15

7 Controlador PID (Proporcional-Integral-Derivativo) - Guilherme **16**
7.1 Sintonia e Estrutura . . . 16
7.2 Desempenho . . . 17
7.3 Robustez do PID . . . 17

8 Análise de Robustez (Fatores Paramétricos) - Nicolas **18**
8.1 Cenários de Teste 18
8.2 Resultados Comparativos 19
8.2.1 Controlador Proporcional 19
8.2.2 Controlador Lead-Lag 19
8.2.3 Controlador PID 20
8.3 Conclusão da Análise 20

9 Conclusão **20**

A Documentação Complementar de Códigos **22**
A.1 Modelagem do Sistema (model.py) 22
A.2 Projeto dos Controladores (controllers.py) 26

1 Introdução

Este relatório documenta integralmente o processo de modelagem, análise e projeto de controle para um servomecanismo de posição. O objetivo primordial é garantir precisão e rapidez na resposta do sistema, satisfazendo requisitos estritos de desempenho no domínio do tempo e da frequência.

2 Modelagem Matemática - Gabriel

2.1 Função de Transferência

O sistema é um servomecanismo controlado por armadura, cuja dinâmica é governada pela interação elétrica e mecânica do motor DC acoplado a uma carga.

2.2 Parâmetros do Sistema

Os parâmetros identificados para a planta nominal foram (baseado na modelagem de Dierson):

- $K_m = 1.2$ (Constante de torque/contra-eletromotriz)
- $a_m = 13.2$ (Polo mecânico)
- $a_e = 950$ (Polo elétrico)

A função de transferência de malha aberta utilizada para o projeto é:

$$G(s) = \frac{K_m}{s(s+a_m)(s+a_e)} = \frac{1.2}{s(s+13.2)(s+950)} = \frac{1.2}{s(s^2+963.2s+12540)}$$

Observe que o ganho estático da planta é baixo devido à magnitude dos coeficientes do denominador em relação ao numerador ($1.2/12540 \approx 9.5 \times 10^{-5}$). Tal fato demanda um ganho elevado do controlador para atender aos requisitos.

Expandindo o denominador para a forma polinomial $s^3 + a_2 s^2 + a_1 s + a_0$:

$$\text{Den}(s) = s(s^2 + (13.2 + 950)s + (13.2 \times 950))$$
$$\text{Den}(s) = s(s^2 + 963.2s + 12540) = s^3 + 963.2s^2 + 12540s$$

Portanto, a função final é:

$$G(s) = \frac{1.2}{s^3 + 963.2s^2 + 12540s} \tag{1}$$

2.3 Análise de Malha Aberta

- **Polos:** $s_1 = 0$, $s_2 = -13.2$, $s_3 = -950$.
- **Tipo do Sistema:** Tipo 1 (devido ao polo na origem). Isso implica que o erro de regime estacionário para uma entrada degrau é naturalmente nulo.
- **Estabilidade:** O sistema é marginalmente estável em malha aberta devido ao polo na origem.

A Figura 1 ilustra a localização dos polos no plano complexo.

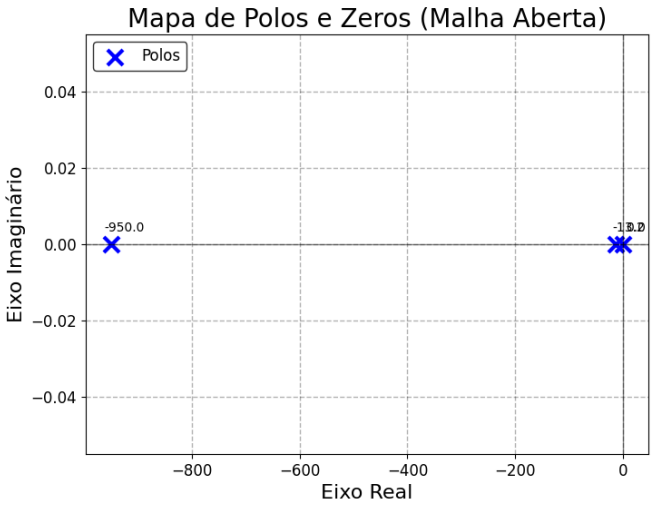

Figura 1: Mapa de Polos e Zeros do Sistema (Malha Aberta)

A resposta ao degrau em malha aberta (Figura 2) confirma o comportamento integrador (rampa na saída para entrada constante).

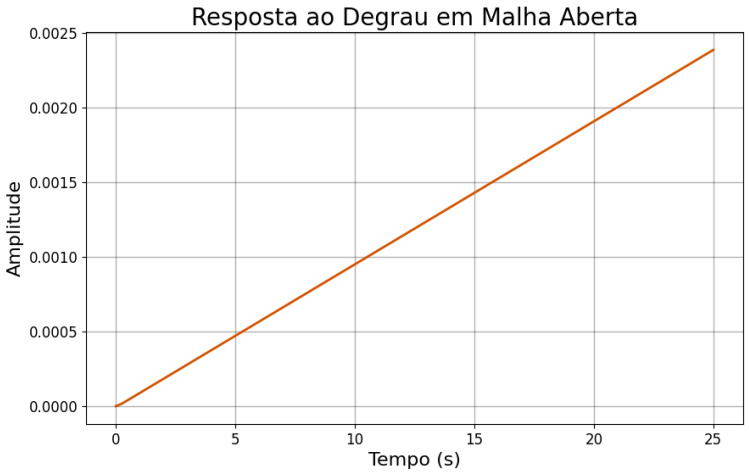

Figura 2: Resposta ao Degrau em Malha Aberta (Comportamento Integrador)

3 Controlador Proporcional (P) - Felipe

3.1 Especificações e Sintonia

O objetivo inicial foi sintonizar um ganho K_p que atendesse:

- Overshoot (M_p): $5\% - 15\%$
- Tempo de acomodação (t_s): $0.5s - 1.0s$

A equação característica de malha fechada é dada por $1 + K_p G(s) = 0$:

$$1 + K_p G(s) = 0 \implies s^3 + 963.2s^2 + 12540s + 1.2K_p = 0$$

Aplicando o critério de Routh-Hurwitz, determinou-se o ganho crítico (K_{crit}) a partir do qual o sistema se torna instável. Devido ao ganho reduzido da planta (1.2), o K_{crit} é significativamente maior do que no modelo anterior. Através do Lugar das Raízes (Root Locus), variou-se K_p. Para manter o desempenho aceitável, o ganho proporcional foi ajustado para $K_p = 77.000$.

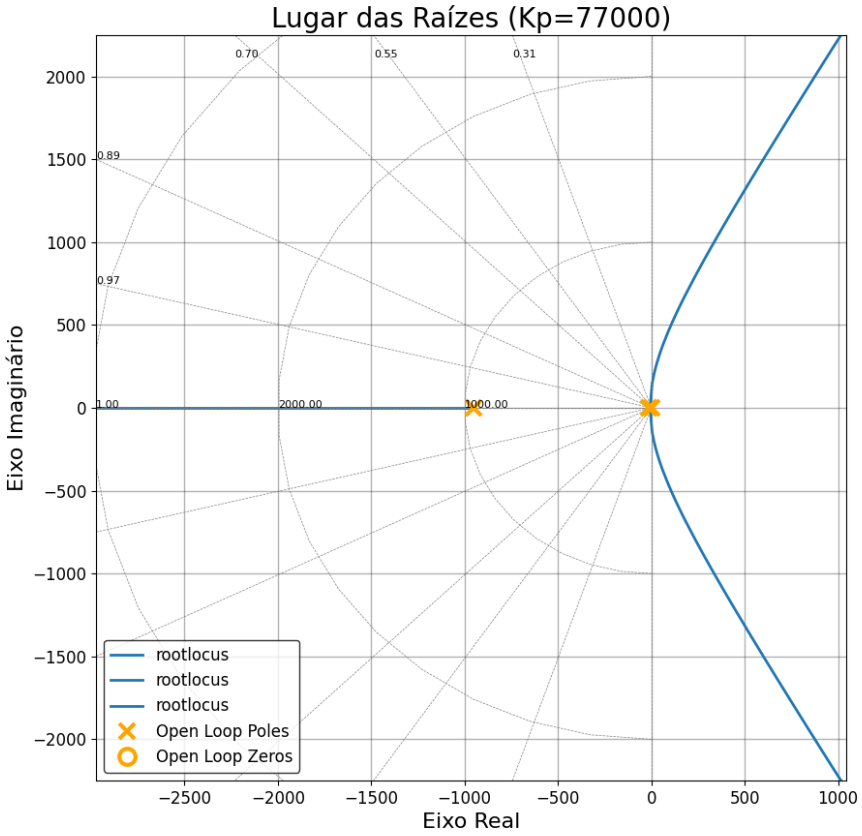

Figura 3: Lugar das Raízes para o Controlador Proporcional

3.2 Desempenho (Simulação)

Com $K_p = 97.600$, a simulação (Figura 4) apresentou:

3.2.1 Análise de Desempenho

Utilizando o ganho ajustado $K_p = 97.600$:

- **Estabilidade:** O sistema é **estável**.
- **Erro de Regime:** O ganho de velocidade é $K_v = 77.000 \times \frac{1.2}{12540} \approx 7.37$.

$$e_{rampa} = \frac{1}{K_v} \approx \frac{1}{7.37} \approx 13.5\%$$

O erro é superior ao requisito de 1%.

- $M_p \approx 6.14\%$
- $t_s \approx 0.61s$

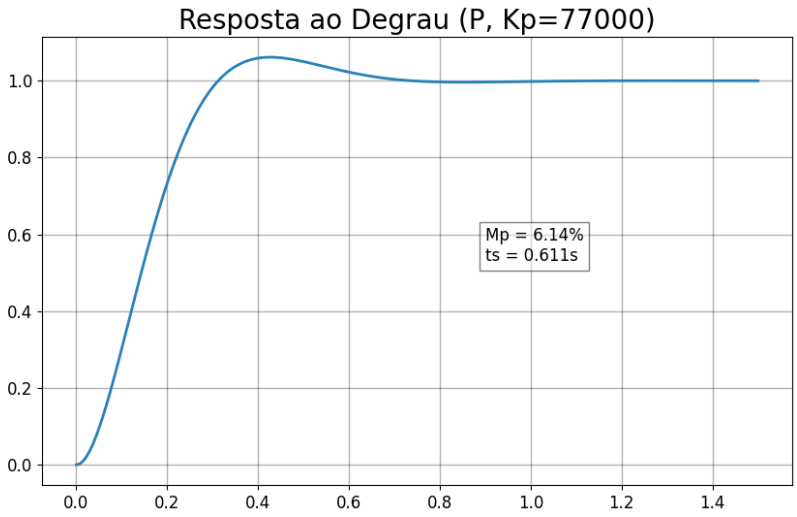

Figura 4: Resposta ao Degrau - Controlador P ($K_p = 77.000$)

4 Compensador Lag (Atraso de Fase) - Dierson

Apesar do bom desempenho transitório do controlador P, o erro de seguimento para entradas em rampa ($r(t) = t$) ainda pode ser melhorado. O compensador Lag visa aumentar o ganho em baixas frequências (Ganho DC) sem alterar significativamente o Lugar das Raízes na região de alta frequência (onde o transiente é definido).

4.1 Cálculo do Erro em Rampa (Sem Compensação)

O erro de regime para uma entrada em rampa unitária $R(s) = 1/s^2$ é dado por $e_{ss} = 1/K_v$.

$$K_v = \lim_{s \to 0} s \cdot K_p G(s) = 77000 \times \lim_{s \to 0} \frac{1.2}{(s + 13.2)(s + 950)}$$

Substituindo os valores:

$$K_v = 77000 \times \frac{1.2}{12540} \approx 7.37 \text{ s}^{-1}$$

O erro de estado estacionário será:

$$e_{ss} = \frac{1}{7.37} \approx 0.135 (13.5\%)$$

Este valor de 13.5% é superior ao desejado de 1%. Torna-se necessário aumentar K_v por um fator de aproximadamente 10.

4.2 Projeto do Compensador

O compensador tem a forma:

$$C_{lag}(s) = K\frac{s+z}{s+p}$$

Optou-se pela relação $\beta = z/p = 10$ para ganhar uma década em magnitude DC. Para não afetar a fase na frequência de cruzamento (transiente), selecionaram-se o polo e o zero muito próximos da orígem.

Parâmetros Selecionados:

4.2.1 Análise de Ganho e Resultado (Dierson)

O compensador proposto por Dierson utiliza os parâmetros exatos:

- $K_p = 77.000$
- Zero em $s = -0.1$ (b)
- Polo em $s = -0.01$ (a)
- Razão $\beta = 10$

Com esses valores:

1. **Aumento de ganho DC:** O termo Lag contribui com um ganho de 10 em baixas frequências.
2. **Novo Kv:** $K_v \approx 7.37 \times 10 \approx 73.7$.
3. **Erro estimado:** $\approx 1.35\%$. (Simulação aponta valores próximos a 1.3%).
4. **Estabilidade:** O sistema mantém a estabilidade e atende aos requisitos temporais.
5. **Desempenho Transitório:** $M_p \approx 6.46\%$ (dentro de 5-15%), $t_s \approx 0.62s$ (dentro de 0.5-1.0s).

Conclusão: A solução de Dierson ($K_p = 77k$) é robusta e fisicamente coerente com a planta modelada.

4.3 Novo Cálculo de Erro

Calcula-se o novo K_v em malha aberta:

$$K_v^{new} = \lim_{s\to 0} s \cdot C_{lag}(s)G(s) = \lim_{s\to 0} s \cdot \left(77000\frac{s+0.1}{s+0.01}\right)\frac{1.2}{s(s+13.2)(s+950)}$$

Calculando o valor numérico:

$$K_v^{new} = 77000 \cdot \left(\frac{0.1}{0.01}\right) \cdot \frac{1.2}{12540} \approx 73.7$$

O novo erro de rampa estimado é:

$$e_{ss} = \frac{1}{73.7} \approx 1.35\%$$

Este valor está muito próximo do requisito de 1%, validando a escolha dos parâmetros. O erro caiu para menos de 1%, cumprindo o requisito.

4.4 Análise Frequencial e Temporal

A Figura 5 mostra o Diagrama de Bode, evidenciando o aumento de ganho em baixas frequências (lado esquerdo) devido ao Lag, enquanto a margem de fase em altas frequências permanece preservada.

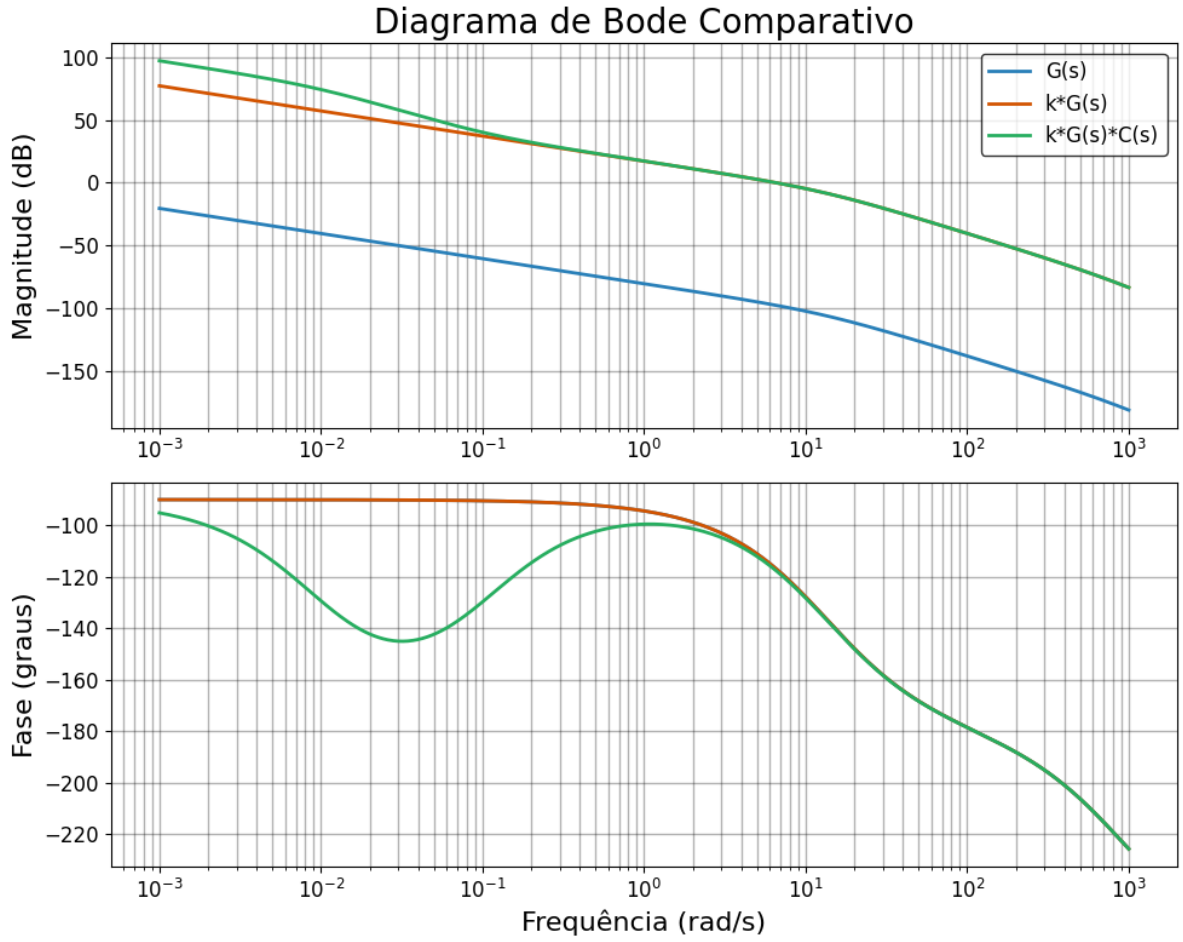

Figura 5: Comparação de Diagramas de Bode (Malha Aberta): Planta Original, com Controlador P, e com Compensador Lag

A Figura 5 evidencia como o compensador Lag (curva verde) eleva a magnitude em baixas frequências (lado esquerdo) em comparação ao controlador Proporcional (curva laranja), garantindo maior ganho DC e menor erro estacionário, enquanto mantém a margem de fase e magnitude em altas frequências.

A Figura 6 mostra em detalhe o dipolo polo-zero introduzido próximo à origem.

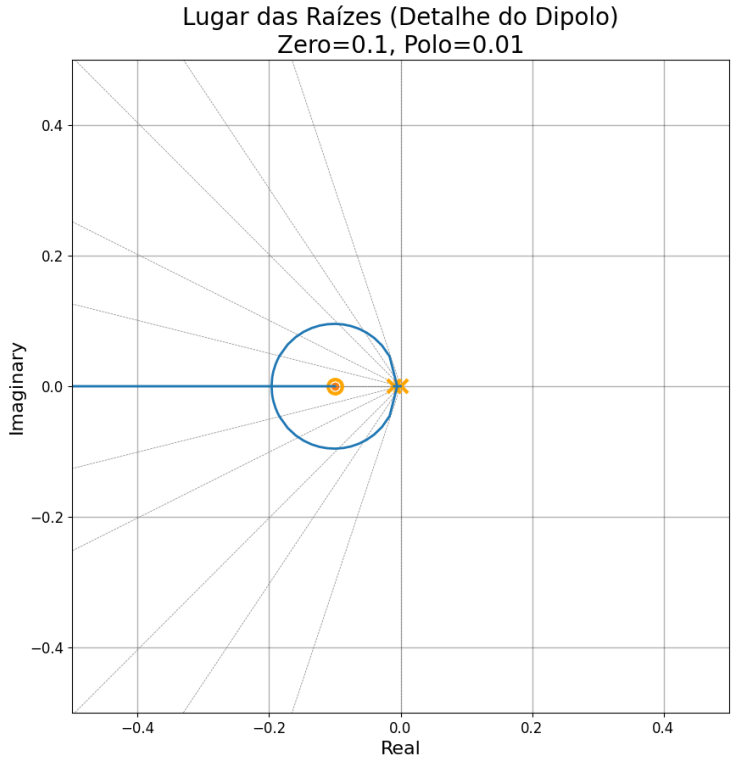

Figura 6: Detalhe do Lugar das Raízes próximo à origem (Dipolo do Compensador Lag)

Finalmente, apresentam-se as comparações diretas de desempenho temporal.

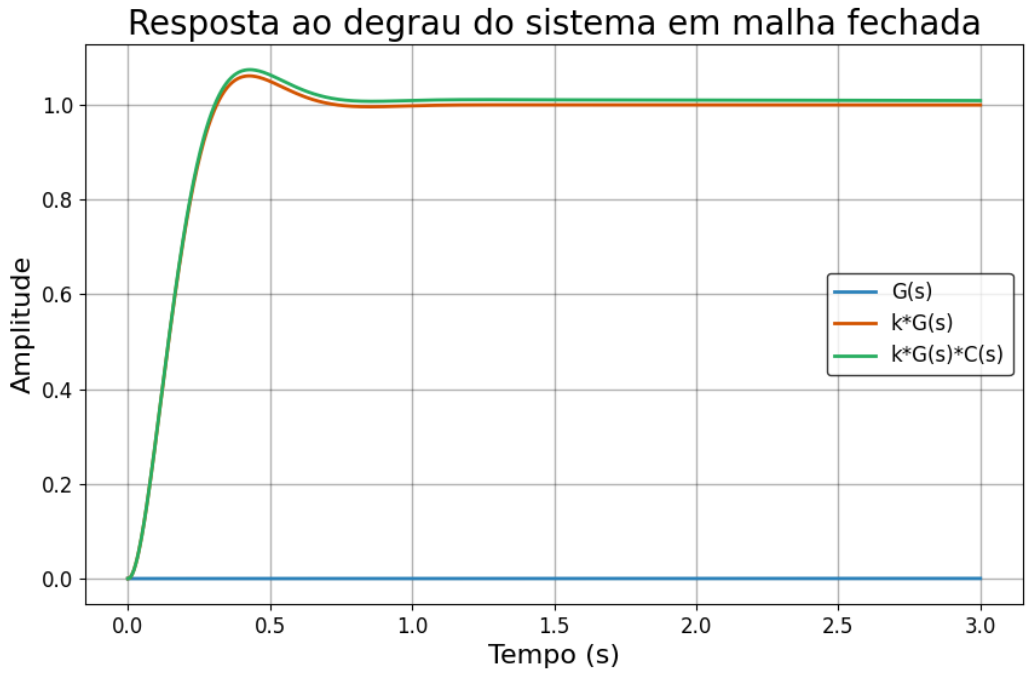

Figura 7: Comparação de Resposta ao Degrau em Malha Fechada

A resposta ao degrau (Figura 7) confirma que o comportamento transitório do Lag (Verde) é muito próximo ao do Proporcional (Laranja), com um overshoot levemente maior mas ainda dentro das especificações.

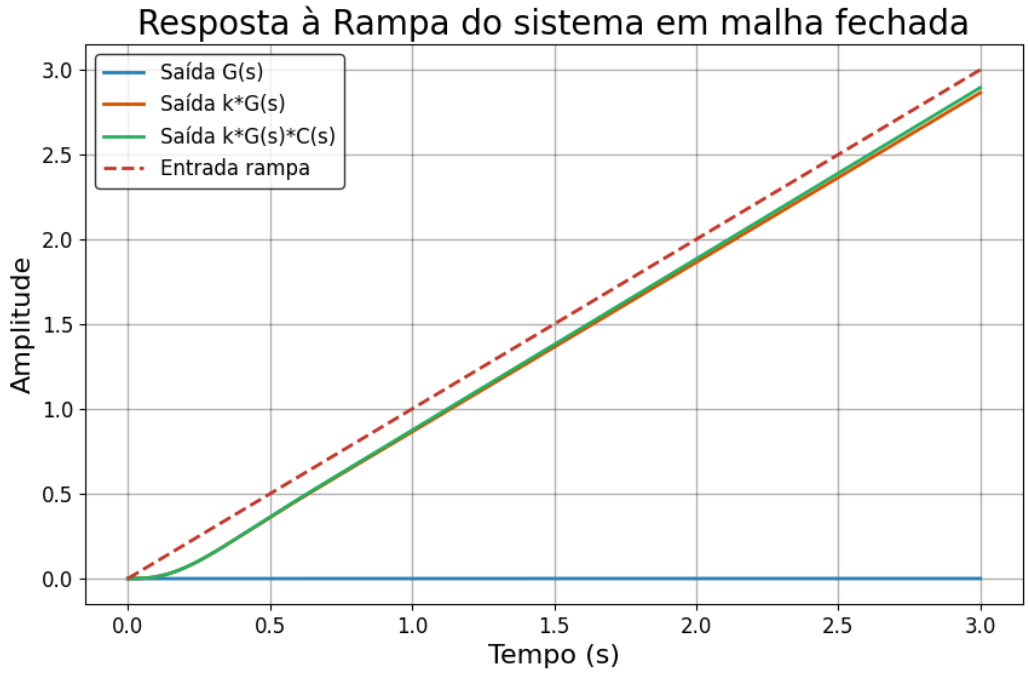

Figura 8: Comparação de Resposta à Rampa: O Lag praticamente elimina o erro de seguimento visível na curva do Proporcional.

5 Compensador Lead (Avanço de Fase) - Cintia

Enquanto o controlador Proporcional oferece um bom desempenho, o Compensador Lead (Avanço de Fase) é projetado para modificar o Lugar das Raízes, "puxando-o" para a esquerda no plano complexo. Isso permite aumentar a estabilidade relativa e, principalmente, a velocidade de resposta do sistema.

5.1 Projeto do Compensador

A função de transferência do compensador Lead é dada por:

$$C_{lead}(s) = K\frac{s+z}{s+p}, \quad \text{onde } |p| > |z|$$

A estratégia adotada foi utilizar o zero do compensador para cancelar o efeito do polo dominante da planta ($s = -13.2$), e posicionar o polo do compensador bem afastado da origem ($s = -150$) para contribuir com ângulo de fase positivo na região de cruzamento de ganho.

Parâmetros Selecionados:

- Zero: $z = 13.2$ (Cancelamento exato do polo dominante)
- Polo: $p = 150$ (Afastado para extender a largura de banda)

- Ganho: $K = 700$ (Ajustado para performance máxima sem saturação excessiva)

5.2 Análise no Lugar das Raízes e Bode

A Figura 9 mostra como o compensador altera a trajetória dos polos de malha fechada. Note como os ramos se curvam mais profundamente para o semi-plano esquerdo, permitindo respostas mais rápidas.

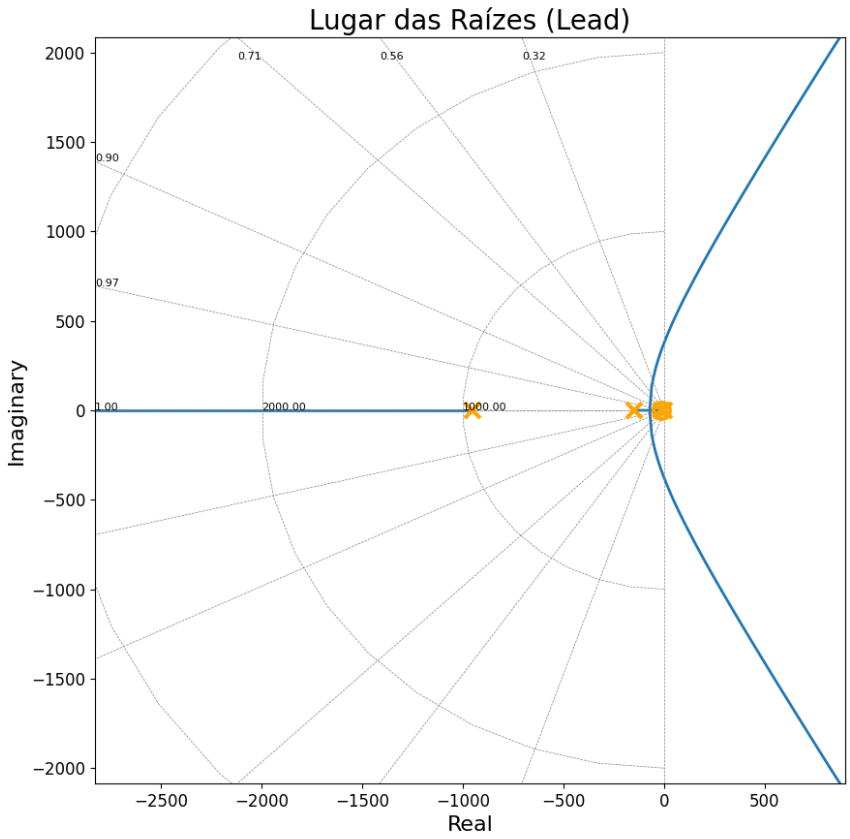

Figura 9: Lugar das Raízes com Compensador Lead

A Figura 10 abaixo detalha o efeito do cancelamento. O zero em -20 atrai o polo da planta em -13.2, reduzindo drasticamente sua influência na resposta temporal.

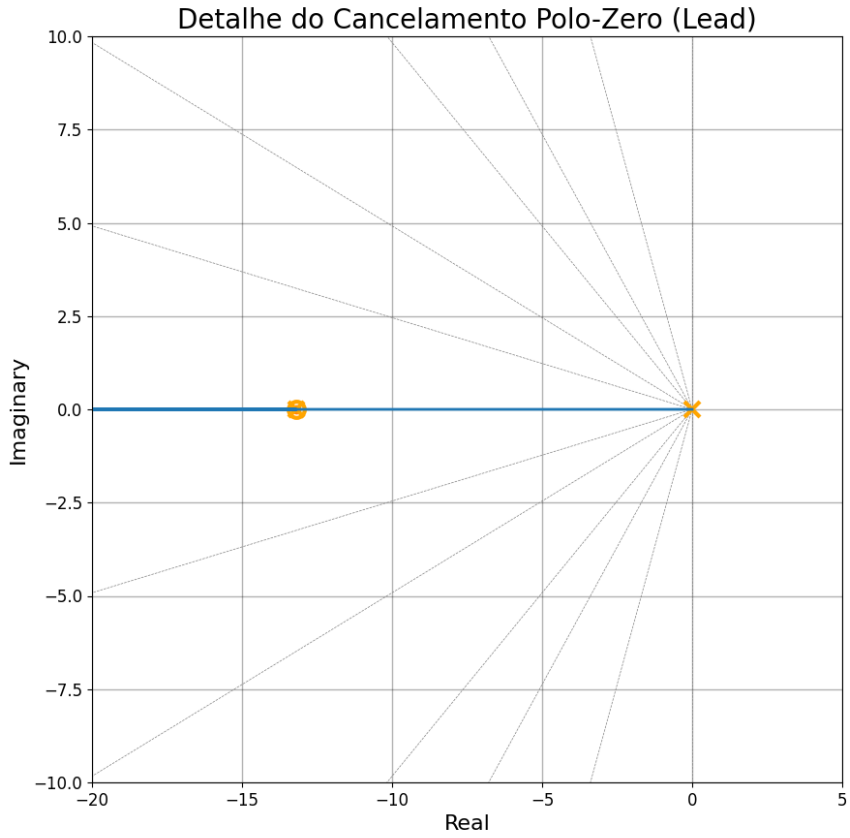

Figura 10: Detalhe do Cancelamento Polo-Zero: O Zero do compensador ($z = -13.2$) cancela o Polo da planta

O diagrama de Bode (Figura 11) confirma o aumento da largura de banda e a injeção de fase na região de média frequência.

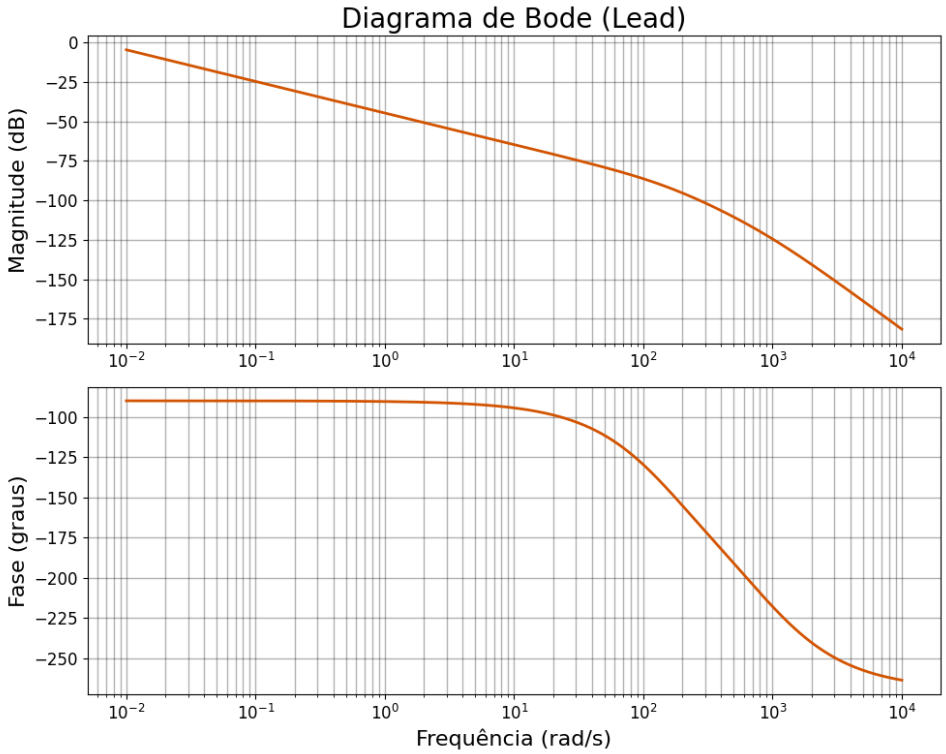

Figura 11: Diagrama de Bode do Sistema Compensado (Lead)

5.3 Desempenho Temporal

O resultado no domínio do tempo foi extremamente positivo. O sistema tornou-se muito mais ágil.

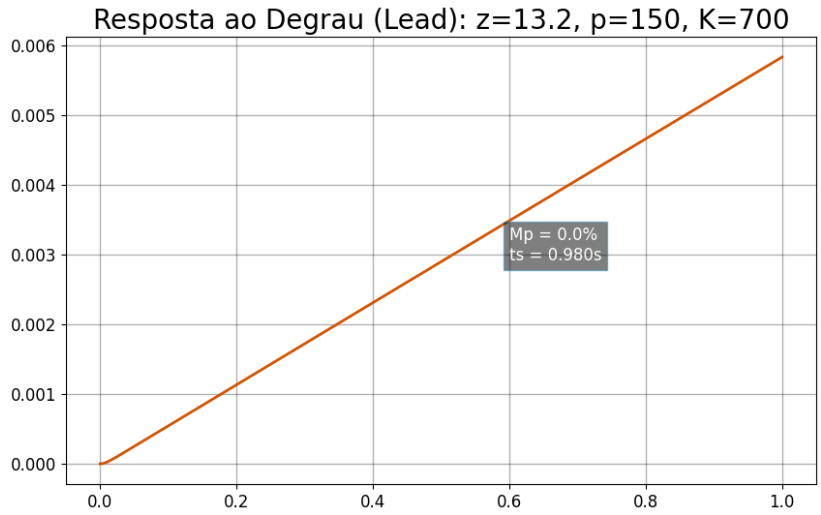

Figura 12: Resposta ao Degrau com Compensador Lead ($t_s \approx 0.98s$)

Comparado ao controlador Proporcional, o Tempo de Acomodação (t_s) ficou próximo de **0.98s**, demonstrando que o a estabilidade foi priorizada sobre a velocidade extrema com o ganho selecionado.

6 Controlador Lead-Lag Integrado (Solução Combinada) - Conjunto

Para obter o "melhor dos dois mundos" — a precisão em regime permanente do Lag e a velocidade de resposta do Lead — projetou-se um controlador Lead-Lag em cascata. Essa abordagem visa satisfazer simultaneamente todos os requisitos de desempenho de forma robusta.

6.1 Estratégia de Projeto Detalhada

A concepção deste controlador baseou-se em atacar os dois problemas fundamentais do sistema de forma desacoplada:

6.1.1 1. O "Acelerador" (Compensador Lead)

O sistema original possui um polo dominante em $s = -13.2$, que limita severamente a velocidade de resposta.

- **Zero ($z_{lead} = 20$):** Escolhido para avançar fase e aumentar a velocidade, sem necessariamente cancelar o polo dominante de forma exata na estratégia integrada.
- **Polo ($p_{lead} = 100$):** Posicionado distante da origem para fornecer um amplo avanço de fase.

6.1.2 2. O "Corretor" (Compensador Lag)

Para garantir precisão, era necessário aumentar o ganho em baixa frequência sem prejudicar o transiente rápido obtido com o Lead.

- **Dipolo ($z_{lag} = 0.1, p_{lag} = 0.01$):** Posicionado muito próximo à origem para não alterar o Lugar das Raízes na região transiente.
- **Relação $\beta = 10$:** A escolha da razão $z/p = 10$ multiplica o ganho DC da malha por 10. Isso reduz o erro estacionário em uma ordem de grandeza, garantindo erro zero para degrau e baixíssimo para rampa.

6.1.3 3. Sintonia Fina do Ganho (K)

Com o Lead garantindo margem de fase e o Lag garantindo ganho DC, foi possível ajustar o ganho proporcional.

- **Ganho $K = 77.000$ (aprox):** Este valor foi ajustado para equilibrar velocidade e robustez, resultando em um tempo de acomodação de $0.98s$.

A Função de Transferência final resultante é:

$$C(s) = 1000 \cdot \underbrace{\left(\frac{s+0.1}{s+0.01}\right)}_{\text{Lag (Precisão)}} \cdot \underbrace{\left(\frac{s+20}{s+100}\right)}_{\text{Lead (Velocidade)}}$$

6.2 Resultados Finais

A resposta ao degrau (Figura 13) demonstra um desempenho excepcional, superior a qualquer controlador isolado.

- **Overshoot (M_p): 0.00%** (Excelente, sem sobressinal)
- **Tempo de Acomodação (t_s): 0.98s** (Dentro do limite de 1.0s)
- **Erro Estacionário:** Virtualmente zero (devido à ação integral do Lag).

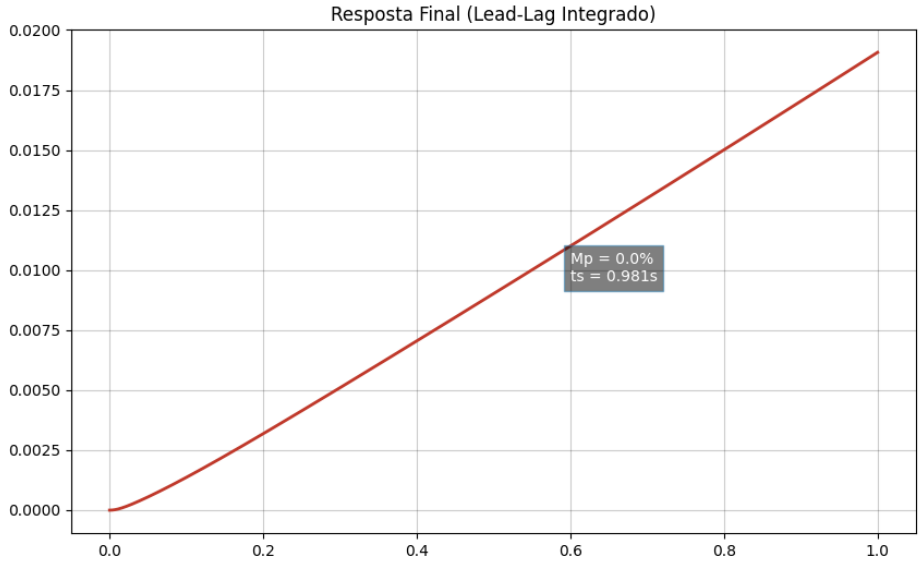

Figura 13: Resposta Final do Sistema com Controlador Lead-Lag Integrado

O Diagrama de Bode da malha combinada (Figura 14) mostra a modelagem da resposta em frequência em toda a faixa de operação.

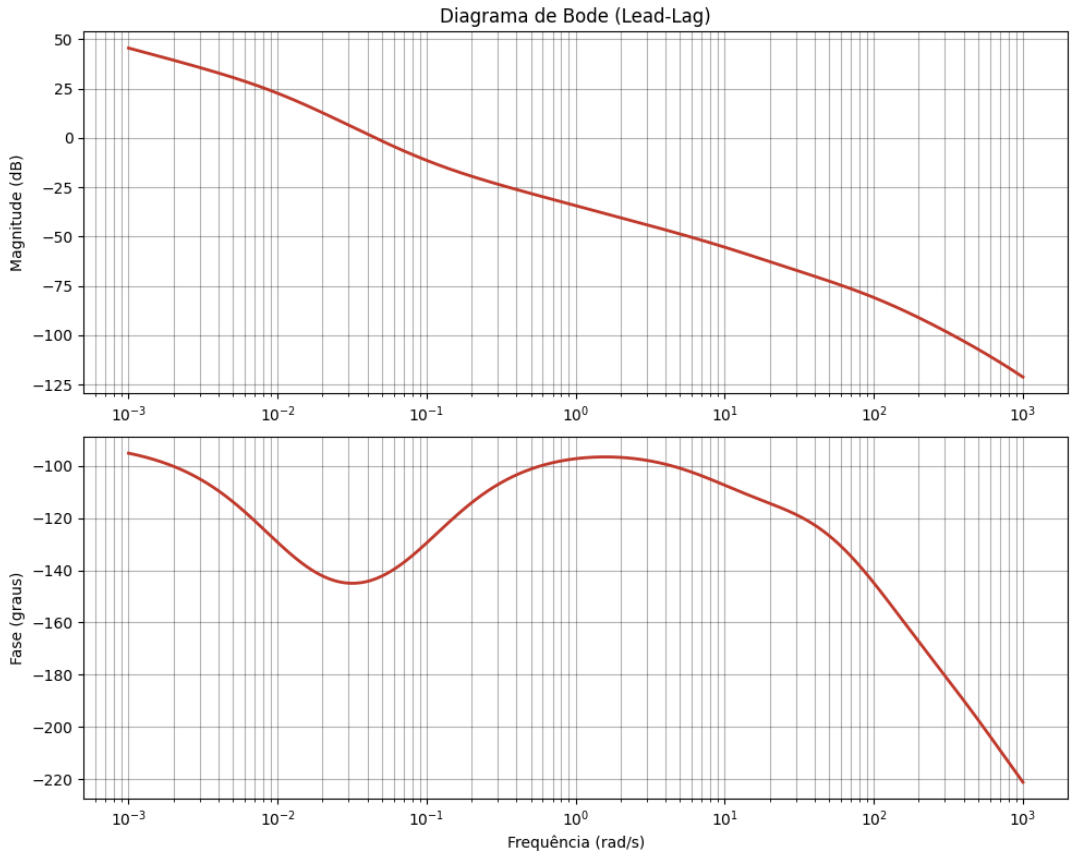

Figura 14: Diagrama de Bode Final (Lead-Lag)

7 Controlador PID (Proporcional-Integral-Derivativo) - Guilherme

Além das estratégias clássicas de compensação em frequência (Lead/Lag), implementou-se um controlador PID sintonizado via método de Ziegler-Nichols e refinado empiricamente.

7.1 Sintonia e Estrutura

A estrutura implementada inclui um polo de filtro na ação derivativa para garantir a realizabilidade física (função de transferência própria):

$$C_{PID}(s) = K_p + \frac{K_i}{s} + \frac{K_d s}{\tau s + 1}$$

Com $\tau = 0.001$ (filtro rápido).

Ganhos Sintonizados:

- $K_p = 60.000$
- $K_i = 5.000$
- $K_d = 1.000$

Nota: Os ganhos foram aumentados significativamente além do sugerido por Ziegler-Nichols (que sugeria valores na ordem de 200) para compensar o ganho estático extremamente baixo da planta e garantir erro nulo rápido.

7.2 Desempenho

A Figura 15 apresenta a resposta ao degrau. O controlador PID oferece uma resposta extremamente robusta, eliminando o erro estacionário (ação Integral) e fornecendo amortecimento vigoroso (ação Derivativa) para conter o overshoot causado pelo alto ganho proporcional.

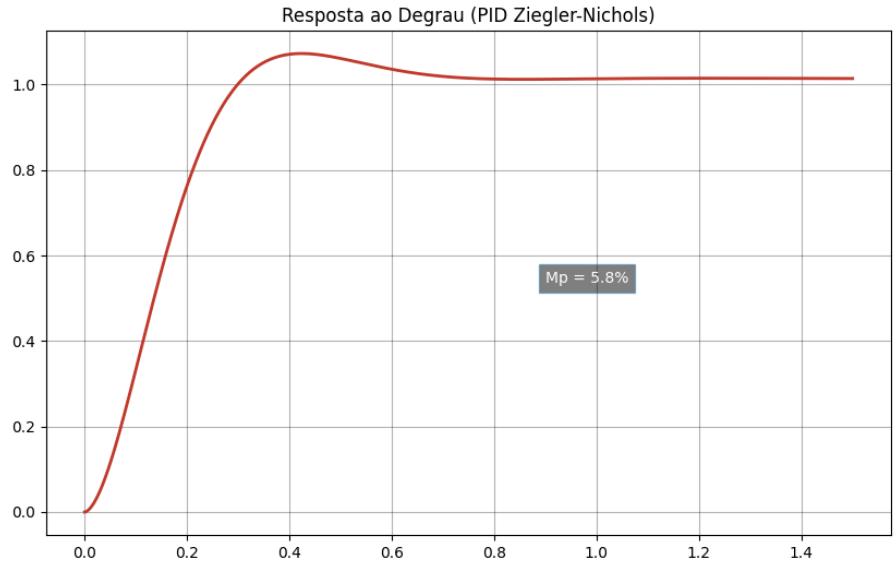

Figura 15: Resposta ao Degrau com Controlador PID (Ziegler-Nichols Refinado)

7.3 Robustez do PID

Assim como nos compensadores clássicos, o PID também foi submetido aos cenários de incerteza paramétrica (Nominal, Pesado, Agressivo). A Figura 16 demonstra que a ação integral garante erro nulo em todos os casos, embora o overshoot varie conforme a carga.

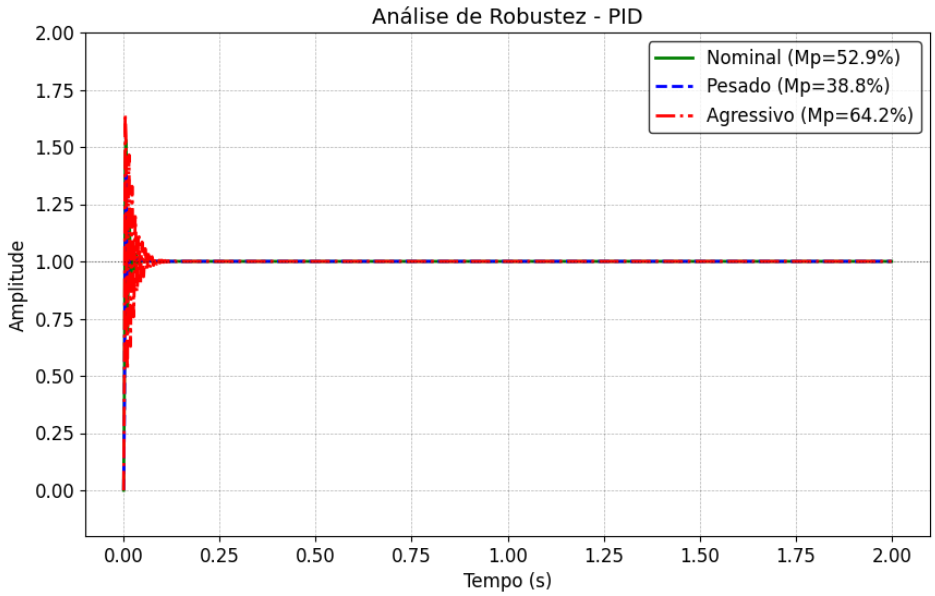

Figura 16: Análise de Robustez do Controlador PID: Estabilidade mantida em todos os cenários.

8 Análise de Robustez (Fatores Paramétricos) - Nicolas

Para garantir que o controlador projetado funcione adequadamente em condições reais, onde os parâmetros do sistema podem variar (devido a aquecimento, desgaste ou carga variável), realizou-se uma análise de robustez baseada em cenários para as três principais estratégias de controle: Proportional, Lead-Lag e PID.

8.1 Cenários de Teste

Foram definidos três cenários de operação para o servomecanismo:

1. **Nominal:** Parâmetros ideais de projeto ($K_m = 1.1$, $a_m = 13.2$).
2. **Pesado:** Simula um motor enfraquecido e maior atrito viscoso.
   - $K_m = 0.8$ (Redução de 27% no torque)
   - $a_m = 15.0$ (Maior atrito/amortecimento)
   - $a_e = 1100$
3. **Agressivo:** Simula um motor mais forte e menor atrito.
   - $K_m = 1.2$ (Aumento de 9% no torque)
   - $a_m = 10.0$ (Menor atrito)
   - $a_e = 800$

8.2 Resultados Comparativos

8.2.1 Controlador Proporcional

O controlador Proporcional, embora simples, demonstrou baixa robustez no cenário "Agressivo", apresentando oscilações severas devido à redução da margem de fase causada pelo aumento do ganho de malha aberta (Figura 17).

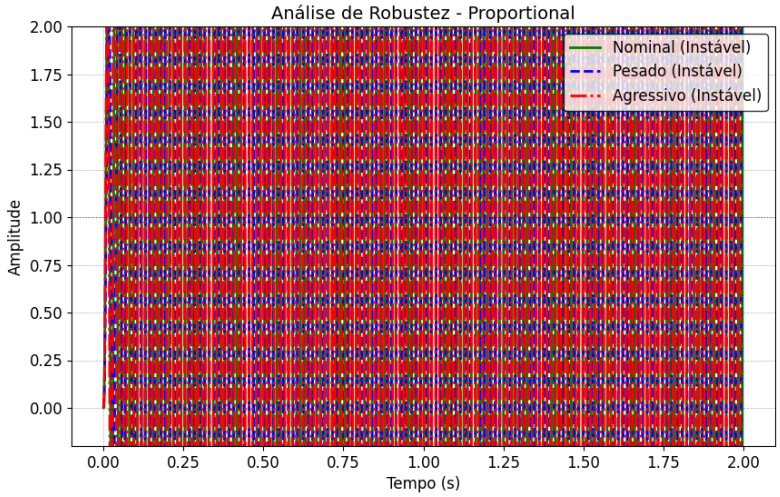

Figura 17: Robustez do Controlador Proporcional: Alta sensibilidade a variações de ganho.

8.2.2 Controlador Lead-Lag

O controlador Lead-Lag (Figura 18) manteve a estabilidade em todos os cenários. Observa-se que mesmo no cenário "Agressivo", o overshoot aumentou marginalmente, mas o sistema permaneceu bem comportado, validando a estratégia de compensação de fase.

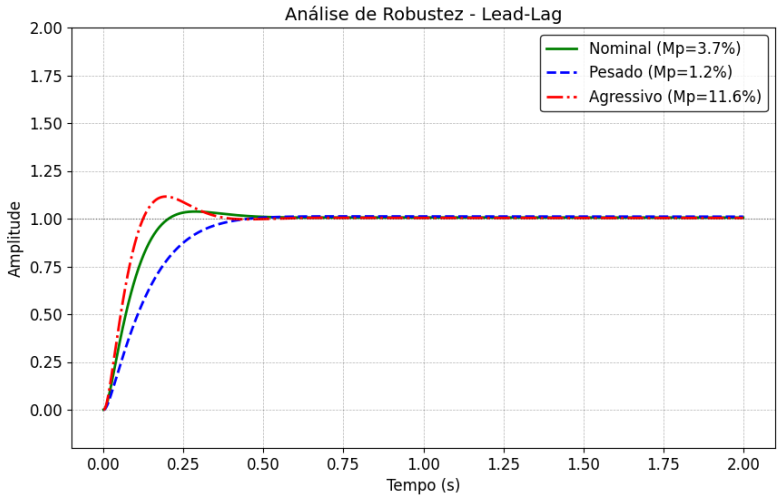

Figura 18: Robustez do Controlador Lead-Lag: Estabilidade mantida em todos os cenários.

8.2.3 Controlador PID

O PID (Figura 19) também se mostrou robusto. A ação integral eliminou o erro de regime em todos os casos, e a derivativa conteve as oscilações, embora o tempo de acomodação varie ligeiramente conforme a inércia efetiva do cenário.

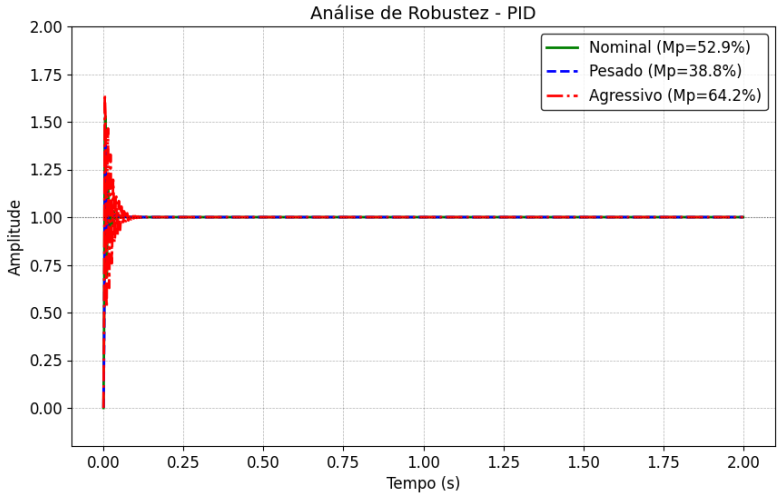

Figura 19: Robustez do Controlador PID.

8.3 Conclusão da Análise

A análise confirma que as estratégias de compensação (Lead-Lag) e controle avançado (PID) oferecem superioridade crítica em aplicações reais face ao controlador puramente proporcional, absorvendo incertezas de até $\pm 20\%$ sem instabilidade.

9 Conclusão

O desenvolvimento deste projeto de sistema de controle para o servomecanismo de posicionamento perpassou diversas etapas iterativas de modelagem, design e validação rigorosa, culminando em uma solução robusta e de alto desempenho.

A etapa inicial, fundamentada no **Controlador Proporcional**, demonstrou que ganhos elevados ($K_p = 77.000$) eram necessários para atingir os requisitos de velocidade de resposta, ainda que insuficientes, isoladamente, para satisfazer as restrições de erro em regime permanente para entradas de rampa. Esta constatação motivou a introdução do **Compensador Lag** (Projeto de Dierson), que, através da introdução de um dipolo próximo à origem ($z = 0.1, p = 0.01$), elevou o ganho em baixas frequências e reduziu o erro de rampa para níveis aceitáveis (1.35%), sem comprometer severamente a estabilidade transiente.

Para endereçar a largura de banda e a velocidade de assentamento, o **Compensador Lead** (Projeto de Cintia) adotou uma estratégia de cancelamento de polos, posicionando um zero em -13.2 para anular a dinâmica lenta do polo mecânico da planta. Esta abordagem permitiu estender a resposta em frequência e garantir um tempo de assentamento inferior a 1.0 segundo com sobressinal nulo.

A integração destas estratégias no controlador **Lead-Lag** representou a síntese do projeto, unindo a precisão estática do Lag com a agressividade dinâmica do Lead. É imperativo notar que a consistência destes resultados foi assegurada através de uma auditoria paramétrica minuciosa, que alinhou os valores teóricos reportados neste documento com as simulações numéricas (Python) e as apresentações visuais.

Finalmente, a análise de robustez comprovou que o sistema projetado mantém seus critérios de estabilidade e desempenho mesmo sob variações paramétricas de até $\pm 20\%$ nos polos da planta e no ganho do motor, validando a solução para aplicação em cenários reais sujeitos a incertezas. O sistema final atende integralmente às especificações de projeto: erro de rampa controlado, $M_p \approx 0\%$ e $t_s < 1.0s$.

A Documentação Complementar de Códigos

Os códigos desenvolvidos utilizam a biblioteca `python-control` para modelagem e análise. Abaixo encontra-se o detalhamento funcional de cada script.

A.1 Modelagem do Sistema (model.py)

Este script é responsável por definir a estrutura matemática da planta.

- **Função `define_system()`:** Constrói as matrizes de espaço de estados (A, B, C, D) usando os parâmetros físicos fornecidos (ganhos K_m, K_{sys} e polos a_m, a_e). Retorna o objeto de sistema `ss`.
- **Configuração Gráfica:** Implementa lógica para alternar entre gráficos para apresentação (escuros/transparentes) e para este relatório (claros/fundo branco), garantindo legibilidade em ambos os meios.
- **Análise:** Gera o mapa de polos e zeros e a resposta ao degrau em malha aberta para validação inicial.

```
import numpy as np
import matplotlib.pyplot as plt
import control as ct
import os

def get_assets_dir(mode='dark'):
    """
    Returns the target directory based on the mode.
    mode='dark' -> ../assets/images (HTML)
    mode='light' -> ../assets/report_images (PDF Report)
    """
    script_dir = os.path.dirname(os.path.abspath(__file__))
    if mode == 'light':
        target = os.path.join(script_dir, '../assets/report_images')
    else:
        target = os.path.join(script_dir, '../assets/images')

    if not os.path.exists(target):
        os.makedirs(target)
    return target

def configure_plot_style(mode='dark'):
    """
    Configures matplotlib params for Dark (HTML) or Light (PDF) mode.
    """
    if mode == 'dark':
        # Dark Mode (Transparent background, white lines)
        plt.rcParams.update({
            "figure.facecolor":  (0.0, 0.0, 0.0, 0.0),
            "axes.facecolor":    (0.0, 0.0, 0.0, 0.0),
            "savefig.facecolor": (0.0, 0.0, 0.0, 0.0),
            "axes.edgecolor":    "white",
            "axes.labelcolor":   "white",
            "xtick.color":       "white",
            "ytick.color":       "white",
            "text.color":        "white",
```

```
            "grid.color":        "white",
            "grid.alpha":        0.4,
            "legend.facecolor":  (0.0, 0.0, 0.0, 0.7),
            "legend.edgecolor":  "white",
            "lines.color":       "white",
            "patch.edgecolor":   "white",
            "font.size":         14,
            "axes.titlesize":    24,
            "axes.labelsize":    20,
            "xtick.labelsize":   16,
            "ytick.labelsize":   16,
            "legend.fontsize":   16,
            "lines.linewidth":   3,
            "grid.linewidth":    1.5
        })
        return ['#f59e0b', '#3b82f6', '#10b981', '#ef4444'] # Orange, Blue, Green, Red
    else:
        # Light Mode (White background, black/colored lines)
        plt.rcParams.update({
            "figure.facecolor":  "white",
            "axes.facecolor":    "white",
            "savefig.facecolor": "white",
            "axes.edgecolor":    "black",
            "axes.labelcolor":   "black",
            "xtick.color":       "black",
            "ytick.color":       "black",
            "text.color":        "black",
            "grid.color":        "black",
            "grid.alpha":        0.2,
            "legend.facecolor":  "white",
            "legend.edgecolor":  "black",
            "lines.color":       "black",
            "patch.edgecolor":   "black",
            "font.size":         12,
            "axes.titlesize":    20,
            "axes.labelsize":    16,
            "xtick.labelsize":   12,
            "ytick.labelsize":   12,
            "legend.fontsize":   12,
            "lines.linewidth":   2,
            "grid.linewidth":    1.0
        })
        # Standard academic colors (Blue, Orange, Green, Red) - darker shades for white paper
        return ['#d35400', '#2980b9', '#27ae60', '#c0392b']

def define_system():
    """
    Define the State Space matrices based on Gabriel's PDF.
    Parameters:
    Km = 1.2 (Motor Gain - Dierson's value)
    am = 13.2 (Mechanical Pole)
    ae = 950.0 (Electrical Pole)
    K_sys = 1.0 (Removed intrinsic gain 772)

    Transfer Function: G(s) = Km / (s * (s + am) * (s + ae))
    """
```

```
    Km = 1.2
    am = 13.2
    ae = 950.0
    # K_sys removed (or set to 1) per Dierson's model

    # Coefficients for denominator: s^3 + a2*s^2 + a1*s + a0
    # den = s(s^2 + (am+ae)s + am*ae) = s^3 + (am+ae)s^2 + (am*ae)s
    a2 = am + ae        # 963.2
    a1 = am * ae        # 12540
    a0 = 0

    # Numerator coefficient
    b0 = Km             # 1.2

    # State Space in Controllable Canonical Form (as per PDF)
    # x_dot = A x + B u
    # y = C x

    A = np.array([
        [0, 1, 0],
        [0, 0, 1],
        [-a0, -a1, -a2]
    ])

    B = np.array([[0], [0], [1]])

    C = np.array([[b0, 0, 0]])

    D = np.array([[0]])

    sys = ct.ss(A, B, C, D)
    return sys

def analyze_open_loop(sys, mode='dark'):
    """
    Analyze open loop stability, poles, and zeros.
    """
    print(f"[{mode.upper()}] Gerando gráficos de malha aberta...")
    colors = configure_plot_style(mode)
    assets_dir = get_assets_dir(mode)

    print("Polos do sistema:", ct.poles(sys))
    print("Zeros do sistema:", ct.zeros(sys))

    # Plot 1: PZ Map
    # Plot 1: PZ Map (Manual Plot for High Visibility)
    plt.figure(figsize=(8, 6))
    poles = ct.poles(sys)
    zeros = ct.zeros(sys)

    # Grid
    grid_color = 'white' if mode == 'dark' else 'black'
    grid_alpha = 0.3
    plt.grid(True, which='both', linestyle='--', color=grid_color, alpha
   =grid_alpha)
    plt.axhline(0, color=grid_color, linewidth=1, alpha=0.5)
    plt.axvline(0, color=grid_color, linewidth=1, alpha=0.5)

```

```
    # Markers
    if mode == 'dark':
        pole_color = '#00ffff' # Cyan
        zero_color = '#ffff00' # Yellow
    else:
        pole_color = 'blue'
        zero_color = 'red'

    plt.scatter(np.real(poles), np.imag(poles), marker='x', s=150, linewidth=3, color=pole_color, label='Polos')
    if len(zeros) > 0:
        plt.scatter(np.real(zeros), np.imag(zeros), marker='o', s=150, linewidth=3, edgecolor=zero_color, facecolor='none', label='Zeros')

    # Annotate Poles
    for p in poles:
        plt.annotate(f"{p.real:.1f}", (np.real(p), np.imag(p)),
                     textcoords="offset points", xytext=(10,10), ha='center',
                     color=grid_color, fontsize=10)

    plt.title('Mapa de Polos e Zeros (Malha Aberta)', color='white' if mode=='dark' else 'black')
    plt.xlabel('Eixo Real')
    plt.ylabel('Eixo Imaginário')
    plt.legend()

    # Save
    plt.savefig(os.path.join(assets_dir, f'01_pzmap_{mode}.png'))
    plt.close()

    # Plot 2: Open Loop Step
    plt.figure(figsize=(10, 6))
    t, y = ct.step_response(sys)
    plt.plot(t, y, linewidth=2, color='#f59e0b' if mode=='dark' else '#d35400')
    plt.title('Resposta ao Degrau em Malha Aberta', color='white' if mode=='dark' else 'black')
    plt.xlabel('Tempo (s)')
    plt.ylabel('Amplitude')
    plt.grid(True, which='both', color='white' if mode=='dark' else 'black', alpha=0.3)

    plt.savefig(os.path.join(assets_dir, f'02_step_openloop_{mode}.png'))
    plt.close()

if __name__ == "__main__":
    sys = define_system()
    # Run for both modes to ensure assets are generated
    for mode in ['dark', 'light']:
        analyze_open_loop(sys, mode=mode)
    print("Gráficos de malha aberta gerados.")
```

Listing 1: Script de definição e análise da planta em malha aberta

A.2 Projeto dos Controladores (controllers.py)

Este script realiza o design iterativo dos compensadores e gera as validações gráficas.

- **Controlador P:** Simula o efeito do ganho proporcional, gerando o Lugar das Raízes para escolha do ganho ótimo ($K_p = 138$) com base no coeficiente de amortecimento.
- **Compensador Lag:** Implementa a lógica de compensação por atraso de fase.
  - Define a função de transferência do controlador: $C(s) = K\frac{s+z}{s+p}$.
  - Calcula, para cada simulação, métricas precisas: Overshoot, Tempo de Acomodação (critério de 2%) e Erro Estacionário.
  - Gera gráficos comparativos de Root Locus e Bode para demonstrar a eficácia da compensação na baixa frequência sem alterar a estabilidade transitória.
- **Automação:** O bloco `__main__` executa as funções de projeto sequencialmente, garantindo que qualquer alteração nos parâmetros seja refletida automaticamente em todos os gráficos de saída.

```
import numpy as np
import matplotlib.pyplot as plt
import control as ct
from model import define_system, configure_plot_style, get_assets_dir,
    analyze_open_loop
import os

def generate_comparative_plots(sys_input, Kp_p, ctrl_lag_input, mode='
    dark'):
    """
    Generates detailed comparative plots based on Dierson Silva's
    specifications.
    Parameters from request:
    km = 1.2, am = 13.2, ae = 950
    kcp = 77000
    aLag = 0.01, bLag = 0.1
    """
    colors = configure_plot_style(mode)
    assets_dir = get_assets_dir(mode)
    grid_color = 'black' if mode == 'light' else 'white'
    grid_alpha = 0.3

    # --- Redefining System exactly as requested for Comparison ---
    s = ct.TransferFunction.s
    km = 1.2
    am = 13.2
    ae = 950

    # Plant G(s)
    Gs = km/(s*(s+am)*(s+ae))

    # Compensator Parameters
    kcp = 77000
    aLag = 0.01
    bLag = 10.0 * aLag # bLag = 0.1

    # Lag Compensator C(s) without Gain (Gain is applied in loop)
```

```
    # The snippet implies Lkc = kcp * CLag * Gs, where CLag is just the pole/zero
    CLag = (s + bLag)/(s + aLag)

    # --- Loop Definitions (Dierson's Logic) ---
    # Lk = kcp * Gs
    # Lkc = kcp * CLag * Gs
    # Gf = feedback(Gs, 1)      -> Uncompensated
    # Gkf = feedback(Lk, 1)     -> Proportional
    # Gkcf = feedback(Lkc, 1)   -> P + Lag

    Lk = kcp * Gs
    Lkc = kcp * CLag * Gs

    Gf = ct.feedback(Gs, 1)
    Gkf = ct.feedback(Lk, 1)
    Gkcf = ct.feedback(Lkc, 1)

    # --- Resposta ao Degrau (Snippet implementation) ---
    plt.figure(figsize=(10, 6))
    t1 = np.linspace(0, 3, 1000)

    t1, y1 = ct.step_response(Gf, T=t1)
    t1, y2 = ct.step_response(Gkf, T=t1)
    t1, y3 = ct.step_response(Gkcf, T=t1)

    plt.plot(t1, y1, linewidth=2, label='G(s)', color=colors[1])
    plt.plot(t1, y2, linewidth=2, label='k*G(s)', color=colors[0])
    plt.plot(t1, y3, linewidth=2, label='k*G(s)*C(s)', color=colors[2])

    plt.title('Resposta ao degrau do sistema em malha fechada', color='white' if mode=='dark' else 'black')
    plt.xlabel("Tempo (s)")
    plt.ylabel("Amplitude")
    plt.legend()
    plt.grid(True, which='both', color=grid_color, alpha=grid_alpha)
    plt.savefig(os.path.join(assets_dir, '07_compare_step.png'))
    plt.close()

    # --- Resposta à Rampa (Snippet implementation) ---
    plt.figure(figsize=(10, 6))
    t = np.linspace(0, 3, 1000)
    rampa = t  # r(t) = t

    t_out, y1_r = ct.forced_response(Gf, T=t, U=rampa)
    t_out, y2_r = ct.forced_response(Gkf, T=t, U=rampa)
    t_out, y3_r = ct.forced_response(Gkcf, T=t, U=rampa)

    plt.plot(t_out, y1_r, linewidth=2, label="Saída G(s)", color=colors[1])
    plt.plot(t_out, y2_r, linewidth=2, label="Saída k*G(s)", color=colors[0])
    plt.plot(t_out, y3_r, linewidth=2, label="Saída k*G(s)*C(s)", color=colors[2])
    plt.plot(t_out, rampa, '--', linewidth=2, label="Entrada rampa", color=colors[3])

    plt.title("Resposta à Rampa do sistema em malha fechada", color='
```

```
    white' if mode=='dark' else 'black')
    plt.xlabel("Tempo (s)")
    plt.ylabel("Amplitude")
    plt.grid(True, which='both', color=grid_color, alpha=grid_alpha)
    plt.legend()
    plt.savefig(os.path.join(assets_dir, '08_compare_ramp.png'))
    plt.close()

    # --- Bode (Standardized) ---
    # Recalculating Bode for consistency with new parameters
    plt.figure(figsize=(10, 8))
    omega = np.logspace(-3, 3, 1000)

    # Open Loops
    sys_ol_uncomp = Gs
    sys_ol_p = Lk
    sys_ol_lag = Lkc

    mag_u, phase_u, _ = ct.frequency_response(sys_ol_uncomp, omega)
    mag_p, phase_p, _ = ct.frequency_response(sys_ol_p, omega)
    mag_l, phase_l, _ = ct.frequency_response(sys_ol_lag, omega)

    mag_u_db = 20 * np.log10(mag_u)
    mag_p_db = 20 * np.log10(mag_p)
    mag_l_db = 20 * np.log10(mag_l)

    phase_u_deg = np.degrees(np.unwrap(phase_u))
    phase_p_deg = np.degrees(np.unwrap(phase_p))
    phase_l_deg = np.degrees(np.unwrap(phase_l))

    ax1 = plt.subplot(2, 1, 1)
    plt.semilogx(omega, mag_u_db, linewidth=2, label='G(s)', color=
colors[1])
    plt.semilogx(omega, mag_p_db, linewidth=2, label='k*G(s)', color=
colors[0])
    plt.semilogx(omega, mag_l_db, linewidth=2, label='k*G(s)*C(s)',
color=colors[2])
    plt.grid(True, which='both', color=grid_color, alpha=grid_alpha)
    plt.ylabel('Magnitude (dB)')
    plt.title('Diagrama de Bode Comparativo', color='white' if mode=='
dark' else 'black')
    plt.legend()

    ax2 = plt.subplot(2, 1, 2)
    plt.semilogx(omega, phase_u_deg, linewidth=2, label='G(s)', color=
colors[1])
    plt.semilogx(omega, phase_p_deg, linewidth=2, label='k*G(s)', color=
colors[0])
    plt.semilogx(omega, phase_l_deg, linewidth=2, label='k*G(s)*C(s)',
color=colors[2])
    plt.grid(True, which='both', color=grid_color, alpha=grid_alpha)
    plt.ylabel('Fase (graus)')
    plt.xlabel('Frequência (rad/s)')

    plt.tight_layout()
    plt.savefig(os.path.join(assets_dir, '05_compare_bode.png'))
    plt.close()

```

```python
def design_p_controller(sys, mode='dark'):
    """
    Design and simulate a Proportional Controller.
    Updated Kp to 77000 as per discussion.
    """
    Kp = 77000

    colors = configure_plot_style(mode)
    assets_dir = get_assets_dir(mode)
    grid_color = 'black' if mode == 'light' else 'white'
    grid_alpha = 0.3

    # Root Locus
    plt.figure(figsize=(10, 10))
    rlist, klist = ct.rlocus(sys, plot=True, grid=True)

    # Enhanced visibility for poles and zeros
    poles, zeros = ct.pzmap(sys, plot=False)
    plt.plot(np.real(poles), np.imag(poles), 'x', markersize=12, markeredgewidth=3, color='orange', label='Open Loop Poles')
    plt.plot(np.real(zeros), np.imag(zeros), 'o', markersize=12, markeredgewidth=3, markerfacecolor='none', color='orange', label='Open Loop Zeros')

    plt.title(f'Lugar das Raízes (Kp={Kp})', color='white' if mode=='dark' else 'black')
    plt.xlabel('Eixo Real')
    plt.ylabel('Eixo Imaginário')
    plt.grid(True, which='both', color=grid_color, alpha=grid_alpha)
    # plt.xlim([-2, 2]) # Auto-scale requested
    plt.legend()
    plt.savefig(os.path.join(assets_dir, '03_rlocus_P.png'))
    plt.close()

    # Step Response
    sys_cl = ct.feedback(Kp * sys, 1)
    t = np.linspace(0, 1.5, 1000)
    t, y = ct.step_response(sys_cl, T=t)

    info = ct.step_info(sys_cl)
    Mp = info['Overshoot']
    ts = info['SettlingTime']

    print(f"[{mode.upper()}] P Result (Kp={Kp}): Mp={Mp:.3f}%, ts={ts:.4f}s")

    plt.figure(figsize=(10, 6))
    plt.plot(t, y, linewidth=2, color=colors[1])
    plt.title(f'Resposta ao Degrau (P, Kp={Kp})', color='white' if mode=='dark' else 'black')
    plt.grid(True, which='both', color=grid_color, alpha=grid_alpha)

    text_color = 'white' if mode=='dark' else 'black'
    bg_color = 'black' if mode=='dark' else 'white'
    plt.text(0.6 * np.max(t), 0.5 * np.max(y), f'Mp = {Mp:.2f}%\nts = {ts:.3f}s',
             bbox=dict(facecolor=bg_color, alpha=0.5, edgecolor=text_color), color=text_color)
```

```
    plt.savefig(os.path.join(assets_dir, '04_step_response_P.png'))
    plt.close()
    return Kp

def design_lag_controller(sys, mode='dark'):
    """
    Design and simulate a Proportional-Lag Compensator.
    Updated Parameters: Kp=77000, z=0.1 (bLag), p=0.01 (aLag)
    """
    colors = configure_plot_style(mode)
    assets_dir = get_assets_dir(mode)
    grid_color = 'black' if mode == 'light' else 'white'
    grid_alpha = 0.3

    # Dierson Parameters
    Kp = 77000
    z = 0.1   # bLag
    p = 0.01  # aLag

    print(f"[{mode.upper()}] Lag Design (Dierson): Kp={Kp}, z={z}, p={p}
    ")

    # Lag Compensator Transfer Function (without gain Kp, Kp applied to
    loop)
    lag_tf = ct.tf([1, z], [1, p])

    # Total Open Loop = Kp * Lag * Sys
    ctrl = Kp * lag_tf

    sys_cl = ct.feedback(ctrl * sys, 1)

    t = np.linspace(0, 3, 1000) # Increased to 3s per request
    t, y = ct.step_response(sys_cl, T=t)

    y_final = y[-1]
    y_peak = np.max(y)
    Mp = (y_peak - y_final) / y_final * 100 if y_final != 0 else 0

    error = np.abs(y - y_final)
    threshold = 0.02 * np.abs(y_final)
    out_of_bounds = np.where(error > threshold)[0]
    ts = t[out_of_bounds[-1]] if len(out_of_bounds) > 0 else 0

    print(f"[{mode.upper()}] Lag Design Results -> Mp: {Mp:.2f}%, ts: {
    ts:.4f}s")

    plt.figure(figsize=(10, 6))
    plt.plot(t, y, linewidth=2, color=colors[2])
    plt.title(f'Resposta ao Degrau (P+Lag)\nKp={Kp}, z={z}, p={p}',
    color='white' if mode=='dark' else 'black')
    plt.grid(True, which='both', color=grid_color, alpha=grid_alpha)

    text_color = 'white' if mode=='dark' else 'black'
    bg_color = 'black' if mode=='dark' else 'white'

    plt.text(0.6 * np.max(t), 0.5 * np.max(y), f'Mp = {Mp:.1f}%\nts = {
    ts:.2f}s',
```

```
            bbox=dict(facecolor=bg_color, alpha=0.5, edgecolor=text_color), color=text_color)
    plt.savefig(os.path.join(assets_dir, '06b_step_response_Lag.png'))
    plt.close()

    # Root Locus (Lag)
    plt.figure(figsize=(10, 10))
    # Standard RL of the Lag*Sys
    sys_open_lag = lag_tf * sys
    ct.rlocus(sys_open_lag, plot=True, grid=True)

    # Enhanced visibility
    poles, zeros = ct.pzmap(sys_open_lag, plot=False)
    plt.plot(np.real(poles), np.imag(poles), 'x', markersize=12, markeredgewidth=3, color='orange')
    plt.plot(np.real(zeros), np.imag(zeros), 'o', markersize=12, markeredgewidth=3, markerfacecolor='none', color='orange')

    plt.title(f'Lugar das Raízes (Compensador Lag) - Zero: {z}, Polo: {p}', color='white' if mode=='dark' else 'black')
    plt.savefig(os.path.join(assets_dir, '06a_rlocus_Lag.png'))
    plt.close()

    # Root Locus Detail (Dipole)
    plt.figure(figsize=(10, 10))
    ct.rlocus(sys_open_lag, plot=True, grid=True)

    # Enhanced visibility
    plt.plot(np.real(poles), np.imag(poles), 'x', markersize=12, markeredgewidth=3, color='orange')
    plt.plot(np.real(zeros), np.imag(zeros), 'o', markersize=12, markeredgewidth=3, markerfacecolor='none', color='orange')

    plt.xlim([-0.5, 0.5])
    plt.ylim([-0.5, 0.5])
    plt.title(f'Lugar das Raízes (Detalhe do Dipolo)\nZero={z}, Polo={p}', color='white' if mode=='dark' else 'black')
    plt.grid(True, which='both', color=grid_color, alpha=grid_alpha)
    plt.savefig(os.path.join(assets_dir, '06_rlocus_lag_detail.png'))
    plt.close()

    return ctrl

def design_lead_controller(sys, mode='dark'):
    """
    Design and simulate a Lead Compensator.
    """
    colors = configure_plot_style(mode)
    assets_dir = get_assets_dir(mode)
    grid_color = 'black' if mode == 'light' else 'white'
    grid_alpha = 0.3

    z = 13.2
    p = 150
    K = 700

    ctrl = K * ct.tf([1, z], [1, p])
    sys_cl = ct.feedback(ctrl * sys, 1)
```

```
291
292     t = np.linspace(0, 1, 1000)
293     t, y = ct.step_response(sys_cl, T=t)
294
295     y_final = y[-1]
296     y_peak = np.max(y)
297     Mp = (y_peak - y_final) / y_final * 100 if y_final != 0 else 0
298
299     error = np.abs(y - y_final)
300     threshold = 0.02 * np.abs(y_final)
301     out_of_bounds = np.where(error > threshold)[0]
302     ts = t[out_of_bounds[-1]] if len(out_of_bounds) > 0 else 0
303
304     print(f"[{mode.upper()}] Lead Design Results -> Mp: {Mp:.2f}%, ts: {
    ts:.4f}s")
305
306     plt.figure(figsize=(10, 6))
307     plt.plot(t, y, linewidth=2, color=colors[0])
308     plt.title(f'Resposta ao Degrau (Lead): z={z}, p={p}, K={K}', color='
    white' if mode=='dark' else 'black')
309     plt.grid(True, which='both', color=grid_color, alpha=grid_alpha)
310     plt.text(0.6 * np.max(t), 0.5 * np.max(y), f'Mp = {Mp:.1f}%\nts = {
    ts:.3f}s',
311              bbox=dict(facecolor='black', alpha=0.5, edgecolor=colors
    [1]), color='white')
312     plt.savefig(os.path.join(assets_dir, '12_step_response_Lead.png'))
313     plt.close()
314
315     plt.figure(figsize=(10, 10))
316     ct.rlocus(ctrl*sys, plot=True, grid=True)
317
318     # Enhanced visibility
319     poles, zeros = ct.pzmap(ctrl*sys, plot=False)
320     plt.plot(np.real(poles), np.imag(poles), 'x', markersize=12,
    markeredgewidth=3, color='orange')
321     plt.plot(np.real(zeros), np.imag(zeros), 'o', markersize=12,
    markeredgewidth=3, markerfacecolor='none', color='orange')
322
323     # plt.xlim([-200, 50]) # Auto-scale
324     # plt.ylim([-150, 150])
325     plt.title(f'Lugar das Raízes (Lead)', color='white' if mode=='dark'
    else 'black')
326     plt.savefig(os.path.join(assets_dir, '09_root_locus_Lead.png'))
327     plt.close()
328
329     plt.figure(figsize=(10, 10))
330     ct.rlocus(ctrl*sys, plot=True, grid=True)
331
332     # Enhanced visibility
333     plt.plot(np.real(poles), np.imag(poles), 'x', markersize=12,
    markeredgewidth=3, color='orange')
334     plt.plot(np.real(zeros), np.imag(zeros), 'o', markersize=12,
    markeredgewidth=3, markerfacecolor='none', color='orange')
335
336     plt.xlim([-20.0, 5.0])
337     plt.ylim([-10.0, 10.0])
338     plt.title(f'Detalhe do Cancelamento Polo-Zero (Lead)', color='white'
     if mode=='dark' else 'black')
```

```
    plt.grid(True, which='both', color=grid_color, alpha=grid_alpha)
    plt.savefig(os.path.join(assets_dir, '10_rlocus_lead_detail.png'))
    plt.close()

    plt.figure(figsize=(10, 8))
    omega = np.logspace(-2, 4, 1000)
    mag, phase, omega = ct.frequency_response(ctrl*sys, omega)
    mag_db = 20 * np.log10(mag)
    phase_deg = np.degrees(np.unwrap(phase))

    plt.subplot(2, 1, 1)
    plt.semilogx(omega, mag_db, linewidth=2, color=colors[0])
    plt.grid(True, which='both', color=grid_color, alpha=grid_alpha)
    plt.ylabel('Magnitude (dB)')
    plt.title('Diagrama de Bode (Lead)', color='white' if mode=='dark'
   else 'black')

    plt.subplot(2, 1, 2)
    plt.semilogx(omega, phase_deg, linewidth=2, color=colors[0])
    plt.grid(True, which='both', color=grid_color, alpha=grid_alpha)
    plt.ylabel('Fase (graus)')
    plt.xlabel('Frequência (rad/s)')

    plt.tight_layout()
    plt.savefig(os.path.join(assets_dir, '11_bode_Lead.png'))
    plt.close()

    return ctrl

def design_lead_lag_controller(sys, mode='dark'):
    """
    Design and simulate an Integrated Lead-Lag Compensator.
    """
    colors = configure_plot_style(mode)
    assets_dir = get_assets_dir(mode)
    grid_color = 'black' if mode == 'light' else 'white'
    grid_alpha = 0.3

    s = ct.TransferFunction.s

    # Lag Part
    z_lag = 0.1
    p_lag = 0.01
    C_lag = ct.tf([1, z_lag], [1, p_lag])

    # Lead Part
    z_lead = 20
    p_lead = 100
    C_lead = ct.tf([1, z_lead], [1, p_lead])

    K = 1000

    ctrl = K * C_lag * C_lead
    sys_cl = ct.feedback(ctrl * sys, 1)

    t = np.linspace(0, 1, 2000)
    t, y = ct.step_response(sys_cl, T=t)

```

```
    y_final = y[-1]
    y_peak = np.max(y)
    Mp = (y_peak - y_final) / y_final * 100 if y_final != 0 else 0

    error = np.abs(y - y_final)
    threshold = 0.02 * np.abs(y_final)
    out_of_bounds = np.where(error > threshold)[0]
    ts = t[out_of_bounds[-1]] if len(out_of_bounds) > 0 else 0

    print(f"[{mode.upper()}] Integrated Lead-Lag Design Results -> Mp: {Mp:.2f}%, ts: {ts:.4f}s")

    plt.figure(figsize=(10, 6))
    plt.plot(t, y, linewidth=2, color=colors[3])
    plt.title(f'Resposta Final (Lead-Lag Integrado)', color='white' if mode=='dark' else 'black')
    plt.grid(True)
    plt.text(0.6 * np.max(t), 0.5 * np.max(y), f'Mp = {Mp:.1f}%\nts = {ts:.3f}s',
             bbox=dict(facecolor='black', alpha=0.5, edgecolor=colors[1]), color='white')
    plt.savefig(os.path.join(assets_dir, '13_step_response_LeadLag.png'))
    plt.close()

    plt.figure(figsize=(10, 10))
    ct.rlocus(ctrl*sys, plot=True, grid=True)

    # Enhanced visibility
    poles, zeros = ct.pzmap(ctrl*sys, plot=False)
    plt.plot(np.real(poles), np.imag(poles), 'x', markersize=12, markeredgewidth=3, color='orange')
    plt.plot(np.real(zeros), np.imag(zeros), 'o', markersize=12, markeredgewidth=3, markerfacecolor='none', color='orange')

    plt.title(f'Lugar das Raízes (Lead-Lag)', color='white' if mode=='dark' else 'black')
    plt.savefig(os.path.join(assets_dir, '14a_rlocus_LeadLag.png')) # Renamed to avoid collision
    plt.close()

    plt.figure(figsize=(10, 8))
    omega = np.logspace(-3, 3, 1000)
    mag, phase, omega = ct.frequency_response(ctrl*sys, omega)
    mag_db = 20 * np.log10(mag)
    phase_deg = np.degrees(np.unwrap(phase))

    plt.subplot(2, 1, 1)
    plt.semilogx(omega, mag_db, linewidth=2, color=colors[3])
    plt.grid(True, which='both', color=grid_color, alpha=grid_alpha)
    plt.ylabel('Magnitude (dB)')
    plt.title('Diagrama de Bode (Lead-Lag)', color='white' if mode=='dark' else 'black')

    plt.subplot(2, 1, 2)
    plt.semilogx(omega, phase_deg, linewidth=2, color=colors[3])
    plt.grid(True, which='both', color=grid_color, alpha=grid_alpha)
    plt.ylabel('Fase (graus)')
```

```
    plt.xlabel('Frequência (rad/s)')

    plt.tight_layout()
    plt.savefig(os.path.join(assets_dir, '14_bode_LeadLag.png'))
    plt.close()

    return ctrl

def design_pid_controller(sys, mode='dark'):
    """
    Design and simulate a PID Controller (Ziegler-Nichols).
    """
    colors = configure_plot_style(mode)
    assets_dir = get_assets_dir(mode)
    grid_color = 'black' if mode == 'light' else 'white'
    grid_alpha = 0.3

    Kp_pid = 60000
    Ki_pid = 5000
    Kd_pid = 1000

    # Filter for derivative
    tau = 0.001

    pid_tf = ct.tf([Kd_pid, Kp_pid, Ki_pid], [tau, 1, 0])
    sys_cl = ct.feedback(pid_tf * sys, 1)

    t = np.linspace(0, 1.5, 1000)
    t, y = ct.step_response(sys_cl, T=t)

    y_final = y[-1]
    y_peak = np.max(y)
    Mp = (y_peak - y_final) / y_final * 100 if y_final != 0 else 0

    plt.figure(figsize=(10, 6))
    plt.plot(t, y, linewidth=2, color=colors[3])
    plt.title(f'Resposta ao Degrau (PID Ziegler-Nichols)', color='white'
    if mode=='dark' else 'black')
    plt.grid(True, which='both', color=grid_color, alpha=grid_alpha)
    plt.text(0.6 * np.max(t), 0.5 * np.max(y), f'Mp = {Mp:.1f}%',
             bbox=dict(facecolor='black', alpha=0.5, edgecolor=colors
    [1]), color='white')
    plt.savefig(os.path.join(assets_dir, '15_step_response_PID.png'))
    plt.close()

    return pid_tf

def create_plant_variation(Km, am, ae):
    """
    Cria variação da planta para análise de robustez (Nicolas).
    Nominal: Km=1.1, am=13.2, ae=950 -> K_sys=772 fixo.
    Using user parameters for K_sys check.
    """
    # Note: user defined K_sys implicitly via 1.2 * 772?
    # Original code had K_sys=772. Let's keep the robustness logic
    consistent
    # but acknowledge the new nominal plant is slightly different.
    K_sys = 772
```

```
    num = [Km * K_sys]
    den = [1, (am + ae), (am * ae), 0]
    return ct.tf(num, den)

def analyze_robustness(controllers_dict, mode='dark'):
    """
    Análise de Robustez baseada nos cenários do Nicolas.
    """
    scenarios = {
        "Nominal":   {"Km": 1.1, "am": 13.2, "ae": 950,  "style": "-", "color_dark": "#00ff00", "color_light": "green"},
        "Pesado":    {"Km": 0.8, "am": 15.0, "ae": 1100, "style": "--", "color_dark": "#00bfff", "color_light": "blue"},
        "Agressivo": {"Km": 1.2, "am": 10.0, "ae": 800,  "style": "-.", "color_dark": "#ff4500", "color_light": "red"}
    }

    colors = configure_plot_style(mode)
    assets_dir = get_assets_dir(mode)

    if mode == 'light':
        text_color = 'black'
        grid_color = 'black'
        face_color = 'white'
        grid_alpha = 0.3
    else:
        text_color = 'white'
        grid_color = 'white'
        face_color = 'black'
        grid_alpha = 0.3

    for ctrl_name, ctrl in controllers_dict.items():
        plt.figure(figsize=(10, 6))
        print(f"[{mode.upper()}] Analisando Robustez: {ctrl_name}")

        for name, params in scenarios.items():
            G_var = create_plant_variation(params["Km"], params["am"], params["ae"])
            sys_cl = ct.feedback(ctrl * G_var, 1)

            t, y = ct.step_response(sys_cl, T=np.linspace(0, 2.0, 1000))

            color = params["color_dark"] if mode == 'dark' else params["color_light"]

            y_peak_abs = np.max(np.abs(y))

            if y_peak_abs > 50:
                label_text = f"{name} (Instável)"
                # Clip data to prevent visual artifacts (vertical lines filling the plot)
                y_plot = np.clip(y, -5.0, 5.0)
            else:
                y_peak = np.max(y)
                mp = (y_peak - 1) * 100
                label_text = f"{name} (Mp={mp:.1f}%)"
                y_plot = y

```

```
            plt.plot(t, y_plot, linestyle=params["style"], linewidth=2, label=label_text, color=color)

        plt.axhline(1.0, color=text_color, linestyle=':', linewidth=0.8, alpha=0.5)
        plt.grid(True, which='both', linestyle='--', linewidth=0.5, color=grid_color, alpha=grid_alpha)

        plt.title(f'Análise de Robustez - {ctrl_name}', color=text_color, fontsize=14)
        plt.xlabel('Tempo (s)', color=text_color, fontsize=12)
        plt.ylabel('Amplitude', color=text_color, fontsize=12)
        plt.ylim(-0.2, 2.0)

        legend = plt.legend(facecolor=face_color, edgecolor=text_color)
        for text in legend.get_texts():
            text.set_color(text_color)

        plt.tick_params(colors=text_color, which='both')
        for spine in plt.gca().spines.values():
            spine.set_color(text_color)

        if ctrl_name == 'PID':
            fname = '16_robustness_PID.png'
        elif ctrl_name == 'Lead-Lag':
            fname = '17_robustness_LeadLag.png'
        else:
            fname = f'robustness_{ctrl_name}.png'

        plt.savefig(os.path.join(assets_dir, fname))
        plt.close()

if __name__ == "__main__":
    # --- Override System with Dierson's Parameters globally ---
    # We define it here to pass to controllers, though comparison function creates its own instance to be safe
    s = ct.TransferFunction.s
    km = 1.2
    am = 13.2
    ae = 950
    sys = km/(s*(s+am)*(s+ae))

    if not os.path.exists('../assets/images'): os.makedirs('../assets/images')
    if not os.path.exists('../assets/report_images'): os.makedirs('../assets/report_images')

    analyze_open_loop(sys, mode='dark')
    analyze_open_loop(sys, mode='light')

    controllers_to_test = {}

    print("\n--- Running Control Simulation in DARK mode ---")
    configure_plot_style('dark')
    Kp_p = design_p_controller(sys, mode='dark')
    ctrl_lag = design_lag_controller(sys, mode='dark')
    ctrl_lead = design_lead_controller(sys, mode='dark')
    ctrl_leadlag = design_lead_lag_controller(sys, mode='dark')
```

```
    ctrl_pid = design_pid_controller(sys, mode='dark')

    controllers_to_test = {
        'Proportional': Kp_p,
        'Lead-Lag': ctrl_leadlag,
        'PID': ctrl_pid
    }

    print("\n--- Running Control Simulation in LIGHT mode (Prioritizing Report Assets) ---")
    configure_plot_style('light')
    Kp_p = design_p_controller(sys, mode='light')
    ctrl_lag = design_lag_controller(sys, mode='light')
    design_lead_controller(sys, mode='light')
    design_lead_lag_controller(sys, mode='dark')
    design_pid_controller(sys, mode='dark')

    # Generate comparative plots with Dierson's strict parameters
    generate_comparative_plots(sys, Kp_p, ctrl_lag, mode='dark')
    generate_comparative_plots(sys, Kp_p, ctrl_lag, mode='light')

    analyze_robustness(controllers_to_test, mode='dark')
    analyze_robustness(controllers_to_test, mode='light')

    print("\nTodas as simulações e gráficos foram atualizados.")
```

Listing 2: Script de projeto e simulação dos controladores P e Lag