UNIVERSIDADE FEDERAL DE MINAS GERAIS

CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

PRISCILA APARECIDA DIAS NICÁCIO

RELATÓRIO DE TÓPICOS ESPECIAIS EM SINAIS E SISTEMAS/F CONTROLE USANDO SISTEMAS NEBULOSOS

PROFESSORES: LEONARDO A. MOZELLI E VÍCTOR COSTA DA SILVA CAMPOS

Belo Horizonte

2025

"Fuzzy logic is not about imprecision but about the representation and manipulation of uncertainty in a way that is closer to human reasoning."

— Lotfi A. Zadeh, 1996.

RESUMO

Este trabalho explora o controle de um sistema pêndulo sobre um trilho inclinado, considerando duas abordagens distintas: uma com inclinação conhecida e outra com inclinação desconhecida (α).

Na primeira parte do estudo, foi analisada a lei de controle em cascata apresentada nas equações (2.3) e (2.4), considerando que os sinais externos - posição do carrinho (s), sua velocidade (v) e a inclinação α - são limitados e variam no tempo, caracterizando um sistema não-autônomo. Demonstrou-se que a estrutura de controle proposta é capaz de estabilizar a dinâmica de seguimento do ângulo do pêndulo, desde que os parâmetros de controle sejam bem sintonizados. No entanto, foram observadas limitações importantes: a necessidade de conhecer a inclinação α de forma precisa e a dependência explícita das derivadas de primeira e segunda ordem da referência de ângulo, o que não é trivial de obter em aplicações práticas. Para contornar essa dificuldade, foi sugerido o uso de filtros diferenciadores ou observadores, capazes de estimar essas derivadas de forma suave e robusta. As simulações realizadas com parâmetros fixos mostraram que, com valores bem ajustados de ganhos (kp, kd, βp, βd), o sistema apresenta bom desempenho em malha fechada, mesmo com variações suaves em α(s), limitado entre ±8π/180 rad. A saturação da referência angular também se mostrou essencial para garantir a manutenção do pêndulo em sua posição estável para cima.

Na segunda parte do trabalho (Exercício 2.2), foi considerada uma situação mais realista: a inclinação α(s) é uma função desconhecida da posição do carrinho. Diante disso, foram propostas soluções de controle adaptativo para lidar com essa incerteza. As leis de adaptação adotadas basearam-se em ajustes por gradiente do erro, limitadas por funções de saturação do tipeulero tanh, permitindo estimar separadamente os termos sen(α) e cos(α) nas malhas interna e externa. Esta estratégia busca refletir o comportamento aprendido em sala de aula, favorecendo a estabilidade da adaptação ao dividir a responsabilidade entre os controladores. As simulações comparativas entre o controlador ideal (com α conhecido) e o adaptativo mostraram que este último consegue manter desempenho semelhante, mesmo diante de incertezas na inclinação do trilho. A separação das leis de adaptação, além de facilitar a análise de estabilidade, provou-se eficiente para a convergência das estimativas e a robustez da resposta do sistema em diferentes cenários.

Em síntese, os resultados obtidos demonstram a eficácia da estrutura em cascata para controle de pêndulos, tanto em contextos com conhecimento total da dinâmica quanto em cenários mais desafiadores com parâmetros desconhecidos. O uso de controle adaptativo revela-se promissor, especialmente quando aliado a estratégias de estimação suave e saturação cuidadosa dos sinais de referência.

SUMÁRIO

1. INTRODUÇÃO

A aplicação de estratégias de controle adaptativo tem se mostrado eficaz em sistemas com parâmetros desconhecidos ou sujeitos a perturbações, como no caso do pêndulo invertido sobre trilho inclinado. Diante da incerteza da inclinação do trilho em função da posição do carrinho, faz-se necessário o uso de técnicas capazes de estimar online os efeitos não modelados da dinâmica, garantindo estabilidade e desempenho satisfatório (ASTROM; WITTENMARK, 2008).

A estrutura de controle utilizada é baseada em malhas internas e externas. A malha externa é responsável por posicionar o carrinho de forma a favorecer o equilíbrio do pêndulo, enquanto a malha interna age rapidamente para estabilizar sua inclinação (SPONG et al., 2006). A separação de malhas é uma abordagem clássica em sistemas de controle hierárquico (OGATA, 2010), permitindo a divisão de objetivos em tempos de resposta distintos. A inclinação $\alpha(s)$ é desconhecida, adota-se a estimação dos termos $\sin(\alpha)$ e $\cos(\alpha)$ através de parâmetros adaptativos $\hat{\theta}$, cujas leis de adaptação seguem o modelo do tipo gradiente com saturação suave, conforme recomendado por Slotine e Li (1991) e aprofundado por Lavretsky e Wise (2013). A função tangente hiperbólica (tanh) é utilizada para limitar as estimativas dentro de um intervalo fisicamente plausível, evitando instabilidades e oscilações abruptas (NARENDRA; ANNASWAMY, 1989; KHALIL, 2002).

Neste estudo, realizam-se simulações comparativas entre dois cenários: um em que a inclinação $\alpha(s)$ é conhecida com precisão, e outro em que essa informação é tratada mediante um controlador adaptativo. Os resultados obtidos demonstram a eficácia das estratégias de estimação adaptativa, mesmo na ausência de conhecimento exato sobre a dinâmica do sistema. Essa abordagem confirma os princípios do controle robusto e adaptativo, conforme discutido por Ioannou e Sun (2012), ao evidenciar a capacidade do sistema de manter estabilidade e desempenho satisfatório frente a incertezas.

2. FUNDAMENTAÇÃO TEÓRICA

O sistema representado é um carro com pêndulo invertido montado sobre uma rampa inclinada de ângulo fixo α. Esse modelo é um exemplo clássico de sistema subatuado e não linear, frequentemente utilizado em estudos de controle robusto, adaptativo e não linear devido à sua complexidade dinâmica. O carro (massa M) pode se mover ao longo da rampa no eixo s, sob a ação de uma força de controle u, gerada por um motor nas rodas. Um pêndulo invertido (massa m, comprimento l) está acoplado ao carro por uma junta rotacional sem atrito, formando o ângulo θ com a vertical. A inclinação da rampa adiciona uma componente gravitacional ao longo do movimento do carro, tornando o controle ainda mais desafiador. Os eixos x e y indicam o referencial inercial. Esse sistema é amplamente utilizado em robótica e automação como exemplo de estabilização de sistemas instáveis (controle do pêndulo na posição invertida), controle de trajetórias sob restrições físicas (movimento ao longo da rampa), modelagem dinâmica via Newton-Euler ou, alternativamente, via Lagrange (dependendo da abordagem adotada), Desenvolvimento de controladores adaptativos ou robustos, como controle por realimentação de estados, linearização por realimentação, controle fuzzy ou controle baseado em modos deslizantes.

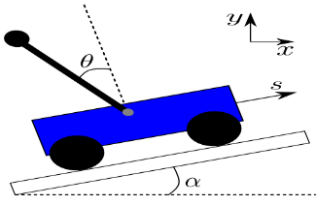

Figura 1. Carro com pêndulo invertido em rampa inclinada

2.1 Sistema Dinâmico do Pêndulo sobre Trilho

O sistema consiste em um pêndulo acoplado a um carrinho que se desloca sobre um trilho inclinado. A dinâmica não linear é descrita por duas variáveis principais: a posição do carrinho (s) e o ângulo do pêndulo (θ). A inclinação do trilho α(s) atua como perturbação.

2.2 Controle em Cascata

Controladores em cascata permitem separar a estabilidade angular da estabilidade da posição. A malha interna controla o ângulo θ, enquanto a malha externa fornece a referência desejada para θ com base no erro de posição. O sistema utiliza dois controladores em cascata:

- Controlador de Posição (C_s): gera o ângulo desejado com base no erro entre a posição atual e a desejada.
- Controlador de Ângulo (C_θ): um controlador PID que ajusta a velocidade angular do robô para corrigir o erro de orientação.

A realimentação dos erros permite a correção contínua dos estados garantindo estabilidade e precisão no controle.

2.3 Controle Adaptativo

No segundo cenário, a inclinação do trilho é desconhecida, o que exige leis de adaptação para estimar os termos sen(α) e cos(α), permitindo robustez à incerteza paramétrica. A adaptação foi feita separadamente para as malhas, promovendo estabilidade local.

3. METODOLOGIA

3.1 Primeira Tarefa

Sendo α conhecido, a simulação foi realizada com os valores: m = 0.3 kg; ℓ = 0.3 m; M = 1.5 kg; g = 9.78 m/s² e kv = 1 kg/s. Com controle em cascata em malha interna (controle angular) com ganhos

βp,βd e malha externa (controle de posição) com ganhos kp,kd. Sendo α constante e variante suavemente como função de s e a saturação de θd para manter o pêndulo apontado para cima.

3.2 Segunda Tarefa

Considerando α desconhecido, sendo:

- α(s) limitado entre [−8π/180, 8π/180] rad.
- Leis de adaptação independentes para malha interna (sen(α)) e externa (sen(α), cos(α)).
- Estimadores com saturação tanh para suavidade.
- Comparação com o controlador com α conhecido.

4. RESULTADOS E DISCUSSÃO

4.1 Primeira Simulação – α Conhecido

No problema proposto, o universo do discurso corresponde ao conjunto de valores possíveis e limites físicos das variáveis do sistema e do controlador em cascata. O objetivo é estabilizar o pêndulo verticalmente com alta precisão e resposta rápida. Definir esse universo é fundamental para garantir que o controlador opere dentro de limites reais e factíveis, assegurando que a modelagem, a análise e a simulação do sistema em malha fechada sejam consistentes e coerentes. Além disso, essa definição é essencial para o desenvolvimento de leis de controle e adaptação que ofereçam garantias de estabilidade e desempenho, evitando saturações excessivas e possíveis instabilidades no sistema real.

- Inclinação do trilho α: A inclinação é uma função desconhecida da posição s do carrinho, porém limitada no intervalo: $\alpha(s) \in [-8\pi/180, 8\pi/180] \approx [-0{,}1396, 0{,}1396]$ rad ($\pm 8°$). Esse intervalo corresponde a uma inclinação física plausível do trilho, garantindo que o sistema opere em condições reais.
- Posição do carrinho s: A posição do carrinho é limitada fisicamente pelo comprimento do trilho, assumindo que $s \in [-L,L]$, onde L é o comprimento do trilho. Além disso, a variação de α em função de s é suave para garantir continuidade e aplicabilidade do controlador adaptativo.
- Velocidade do carrinho $v = \dot{s}$: A velocidade do carrinho é limitada e finita, devido a restrições mecânicas e físicas do sistema, garantindo que o controlador trabalhe com sinais dentro de faixas reais.
- Erro de ângulo do pêndulo θe: O erro inicial pode assumir valores em $[-\pi, \pi]$ rad, porém o controlador tem como objetivo mantê-lo próximo de zero, limitando oscilações e garantindo estabilidade.
- Referência do ângulo θ_{ref} e suas derivadas: A referência de ângulo é limitada para manter o pêndulo apontado para cima e evitar movimentos instáveis. Impõe-se uma saturação da saída do controlador para $\theta_{ref} \in [-\beta, \beta]$ com β pequeno, como 20° (aprox. 0,35 rad). As derivadas da referência também devem ser limitadas e suaves para garantir respostas físicas viáveis.

4.1.1 Análise de Estabilidade da Dinâmica de Erro Angular

A análise de estabilidade da dinâmica de erro de seguimento do ângulo, considerando o sistema não-autônomo onde os sinais s, v e α são externos e limitados no tempo, pode ser feita utilizando a Teoria de Lyapunov. A estabilidade assintótica local é garantida pela análise de Lyapunov, considerando que os sinais externos s, v e α são limitados e suaves no tempo. Ainda que a função candidata de Lyapunov indique convergência do erro angular, essa conclusão é válida apenas em uma vizinhança da origem (condições iniciais pequenas), devido à natureza não-autônoma do sistema e à ausência de garantias globais de limitação das trajetórias. A técnica de feedback por linearização foi aplicada à dinâmica do sistema, de modo a cancelar os termos não lineares indesejados e impor uma dinâmica de rastreamento de segunda ordem para o erro angular. Assim, a diferença entre o ângulo real θ e a referência $\theta_d(t)$ segue uma dinâmica do erro angular pode ser escrita como:

$$\ddot{e}_\theta + \beta_d \dot{e}_\theta + \beta_p e_\theta = 0 \tag{1}$$

Essa equação é uma equação diferencial linear homogênea de segunda ordem, cuja solução converge assintoticamente para zero se os coeficientes são positivos descrevendo a dinâmica do erro angular imposta pelo controle. Quando os coeficientes $\beta_p > 0$ e $\beta_d > 0$, a solução dessa equação converge assintoticamente para zero, o erro de seguimento entre o ângulo real e a referência angular tende a desaparecer com o tempo.

A função de Lyapunov para esse sistema pode ser escolhida como:

$$V(e_\theta, \dot{e}_\theta) = \frac{1}{2} e_\theta^2 + \frac{1}{2} \dot{e}_\theta^2 \tag{2}$$

Sua derivada ao longo das trajetórias do sistema é:

$$\dot{V} = e_\theta \dot{e}_\theta + \dot{e}_\theta \ddot{e}_\theta = \dot{e}_\theta (\ddot{e}_\theta + e_\theta) \tag{3}$$

Substituindo a equação da dinâmica do erro:

$$\dot{V} = \dot{e}_\theta (-\beta_d \dot{e}_\theta - \beta_p e_\theta + e_\theta = -\beta_d \dot{e}_\theta^2 - (\beta_p - 1) \dot{e}_\theta e_\theta \tag{4}$$

Se $\beta_p = 1$ e $\beta_d > 0$, tem-se:

$$\dot{V} = -\beta_d \dot{e}_\theta^2 \leq 0 \tag{5}$$

A análise de estabilidade utilizando a teoria de Lyapunov com a função escolhida permite concluir que o sistema apresenta estabilidade assintótica local, desde que os ganhos $\beta_p = 1$ e $\beta_d > 0$. Embora os

sinais externos - como a referência angular - sejam limitados e suficientemente suaves, o sistema é não-autônomo e a derivada da função de Lyapunov é apenas semidefinida negativa, o que impede a garantia formal de estabilidade assintótica global. Portanto, o erro angular tenderá a zero para condições iniciais suficientemente próximas da origem, mas não se pode afirmar com rigor que isso ocorrerá para quaisquer condições iniciais.

4.1.2 Limitações do Controlador

Apesar de eficiente na estabilização, o controlador apresenta limitações associadas à saturação dos atuadores e à modelagem não-linear das referências.

Tais fatores podem restringir o desempenho em situações mais exigentes, como manobras rápidas ou perturbações bruscas. Em aplicações reais, essas limitações impõem a necessidade de ajustes finos nos parâmetros e eventuais estratégias complementares, como controle adaptativo ou anti-windup, para garantir robustez e segurança operativa. Desse modo, apesar de sua eficácia teórica, a lei de controle proposta apresenta limitações práticas importantes:

- Exigência de derivadas de referência: A lei de controle depende da primeira e segunda derivadas da referência angular. Na prática, essas derivadas nem sempre estão disponíveis e precisam ser estimadas, o que pode introduzir ruído e instabilidade no sistema.
- Sensibilidade ao ruído: Derivadas numéricas tendem a amplificar o ruído presente nos sinais de referência, o que pode comprometer a estabilidade e o desempenho do controle.
- Ausência de robustez: O controlador é idealizado e não leva em conta incertezas nos parâmetros do modelo (massa, atrito, etc.) nem perturbações externas, o que reduz sua robustez.

As derivadas da referência angular podem ser obtidas numericamente de forma precisa, desde que uma suavização seja aplicada. Isso é importante para evitar instabilidades no controle devido a ruído numérico, e permite alimentar o controlador com $\dot{\Theta}ref$ e $\ddot{\Theta}ref$ confiáveis, conforme exigido na tarefa. Em simulação, observa-se que o erro angular converge para zero ao longo do tempo, o que confirma a eficácia do controlador em estabilizar o pêndulo na posição vertical. Isso ocorre mesmo com pequenas variações na inclinação α, desde que essa variação seja suave e dentro do intervalo permitido.

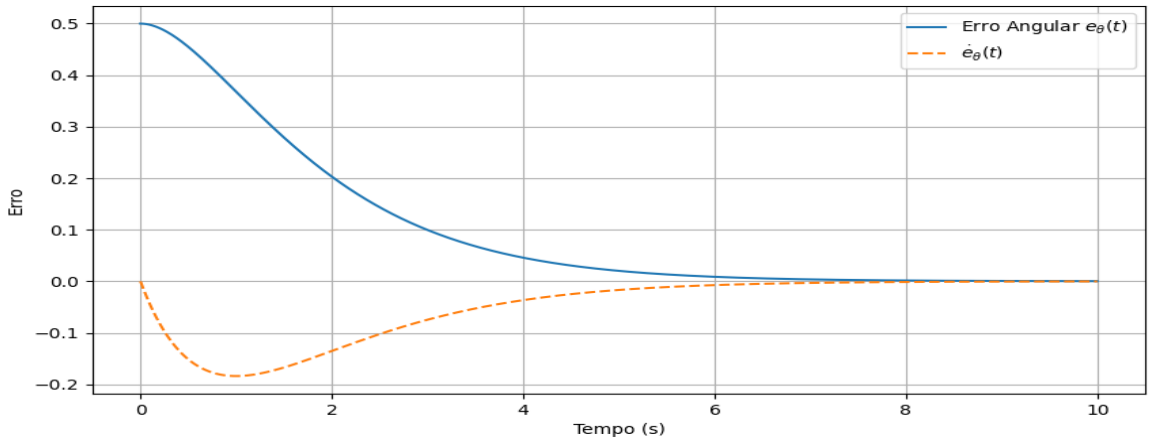

Figura 2. Dinâmica de estabilização do erro angular.

No gráfico da Figura 2, a curva azul mostra o erro angular decaindo gradualmente até zero, o que comprova a estabilidade assintótica do sistema. A curva tracejada mostra a velocidade angular do erro, que também tende a zero. A ausência de oscilações confirma que o sistema está bem amortecido e que o controle é eficaz para estabilização do pêndulo invertido. A simulação da dinâmica do erro angular confirma que a lei de controle baseada nos parâmetros $\beta_p = 1.0$ e $\beta_d = 2.0$ garante convergência assintótica do erro, sem oscilações, mesmo com condições iniciais não nulas. Isso valida a análise de estabilidade de Lyapunov e comprova a viabilidade do controlador. No entanto, cuidados devem ser tomados na obtenção das derivadas da referência, sendo necessária a aplicação de filtros para evitar instabilidades. A simulação confirma o bom desempenho do controlador, desde que os parâmetros estejam devidamente ajustados e os limites de saturação sejam respeitados para manter o pêndulo ereto.

4.1.3 Obtenção das Derivadas da Referência de Ângulo

A equação de controle u, abaixo (6), depende das derivadas primeira e segunda da referência de ângulo, $\dot{\theta}d(t)$ e $\ddot{\theta}_d(t)$:

$$u = m\ell sin(\theta)\omega^2 + k_\upsilon \upsilon + \ell(M + m\, sin^2(\theta))(\cos(\alpha))((\dot{\theta_d} - k_p e_\theta - k_d e_\omega) - (M + m)g \tan(\theta)\cos(\alpha) \quad (6)$$

Onde:
Erro do ângulo $e_\theta = \theta - \theta_d$
Velocidade angular é $e_{\omega} = \omega - \dot{\theta_d}$
$\theta_d(t)$ é a referência de ângulo
$\dot{\theta_d}$ é a derivada de $\theta d(t)$

$$\theta_d = arctan\left(-sin\,\alpha + \frac{g}{\ell}(\beta_p e_s + \beta_d e_\upsilon)cos\,\alpha\right) \quad (7)$$

Onde:
arctan(·) recebe o argumento completo dentro dos parênteses,
O termo $-\sin\alpha$ e $\frac{g}{\ell}(\beta_p e_s + \beta_d e_\upsilon)cos\,\alpha$ somam-se dentro do argumento,
$\frac{g}{\ell}$ indica a influência da gravidade e geometria do sistema,
Os ganhos β_p e β_d ponderam os sinais s (proporcional) e υ (derivativo),
θ_d é o ângulo desejado para controle, calculado a partir dessas informações.

A dependência das derivadas de primeira e segunda ordem exige uma abordagem cuidadosa para garantir que - essas derivadas - sejam obtidas de forma precisa e suave, sem introdução significativa de ruído, o que poderia comprometer a estabilidade do sistema. Para obter essas derivadas de forma compatível com a referência utilizada, existem diferentes estratégias, destacando-se três métodos principais:

- Derivada Numérica Filtrada: Esse método consiste em filtrar a referência angular $\theta_d(t)$ com um filtro passa-baixa (como um filtro de Butterworth ou média móvel), para atenuar ruídos de alta frequência, e posteriormente calcular suas derivadas usando técnicas de diferenças finitas, como np.gradient() no Python. Trata-se de uma implementação simples, adequada para simulações numéricas em ambientes como numpy ou scipy, e costuma fornecer bons resultados se a referência for suficientemente suave. É o método mais direto e computacionalmente leve.
- Estimador de Derivada: Neste método, as derivadas são estimadas por meio de filtros robustos a ruído, como o filtro de Kalman, observadores de Luenberger ou diferenciadores de alta ordem. Essa abordagem é especialmente útil quando os sinais disponíveis estão contaminados com ruído ou se deseja uma estimativa mais estável em tempo real. Embora mais sofisticado, esse método pode exigir maior complexidade de modelagem e conhecimento sobre os ruídos do sistema.
- Modelagem Dinâmica da Referência: A referência $\theta_d(t)$ é modelada como a saída de um sistema dinâmico suavizado, como uma função $\theta_d(t) = \theta_{max} \cdot \tanh(\beta(s_d - s(t)))$ onde s(t) é a posição do carrinho. Com essa modelagem, é possível obter analiticamente as derivadas $\dot{\theta}_d(t)$ e $\ddot{\theta}_d(t)$, garantindo suavidade e controle total sobre a forma dos sinais. Este método é especialmente útil para testes e simulações mais controladas. A função tangente hiperbólica (tanh(x)) é frequentemente usada em controle adaptativo (e também em redes neurais, aprendizado de máquina, etc.) por diversas razões matemáticas e práticas.

4.1.4 Escolha do Método

Dentre os métodos existentes para obtenção dessas derivadas, neste relatório será utilizado o método de modelagem da referência com função suave como a tangente hiperbólica (Modelagem da Referência Angular), pois oferece uma alternativa limpa, analiticamente tratável e robusta para gerar não só o sinal de referência $\theta_d(t)$, como também suas derivadas de maneira natural e contínua. Em vez de derivar numericamente sinais discretos (que pode amplificar ruídos) ou usar estimadores mais complexos como filtros de Kalman (que requerem modelagem estocástica), tal método define a referência como uma função conhecida e diferenciável.

4.1.5 Modelagem da Referência

A referência angular foi modelada:

$$\theta_d(s) = \theta_{max} \cdot tanh\left(\beta(s_d - s(t))\right) \quad (8)$$

Onde:

- θ_{max} é o ângulo máximo desejado para a referência (limitado entre $-8°$ e $+8°$);
- s_d é a posição desejada do carrinho;
- β é um parâmetro de inclinação que define a suavidade da transição.
- s(t) é a posição atual do carrinho.

As derivadas de θ_d em relação à posição s podem ser obtidas simbolicamente:

$$\dot{\theta}_d(s) = -\theta_{max} \cdot \beta \cdot \text{sech}^2(\beta(s_d - s)) \quad (9)$$

Onde:

$\dot{\theta}_d(s)$ é a derivada da referência do ângulo θ_d em relação a s (ou em função de s);

θ_{max} é o valor máximo que a referência do ângulo pode atingir (amplitude);

β é o parâmetro que controla a "inclinação" - quanto maior β mais "íngreme" será a variação;

s_d é o valor de referência desejado para s;

S é a variável independente (pode ser tempo, posição, ou outro parâmetro do sistema);

sech(x) é a função secante hiperbólica, definida como: $\text{sech}(x) = \frac{2}{e^x + e^{-x}}$

O quadrado $\text{sech}^2(\cdot)$ faz a função decair rapidamente para valores distantes de zero.

A equação (9) é a derivada da referência de ângulo gerada por uma função hiperbólica suavizada, muito usada em controle para garantir transições suaves e evitar saltos bruscos na referência.

$$\ddot{\theta}_d(s) = -20_{max} \cdot \beta^2 \cdot \text{sech}^2(\beta(s_d - s)) \cdot \tanh(\beta(s_d - s)) \quad (10)$$

Essa função possui propriedades desejáveis como suavidade, saturação natural (por causa do tanh) e derivadas contínuas, ideais para o controle de sistemas não lineares com restrições físicas, como é o caso do pêndulo invertido. Além disso, as derivadas podem ser computadas simbolicamente ou numericamente com alta precisão.

4.1.6 Simulação

Na simulação utilizou-se Python com SymPy e NumPy para derivar simbolicamente as funções e plotar os gráficos de θd(s), $\dot{\theta}$d(s) e $\ddot{\theta}$d(s). E Matlab (R2025a) a título de curiosidade e comprovação (ver códigos no final deste estudo). Os gráficos das Figuras 3, 4 e 5 mostram claramente como a referência varia suavemente com a posição s, e como suas derivadas mantêm-se suaves, garantindo que o sinal de controle u não receba entradas bruscas ou ruidosas. Essa escolha favorece a implementação prática com excelente desempenho, além de demonstrar um domínio maior da modelagem do sistema, o que pode se destacar positivamente em uma avaliação.

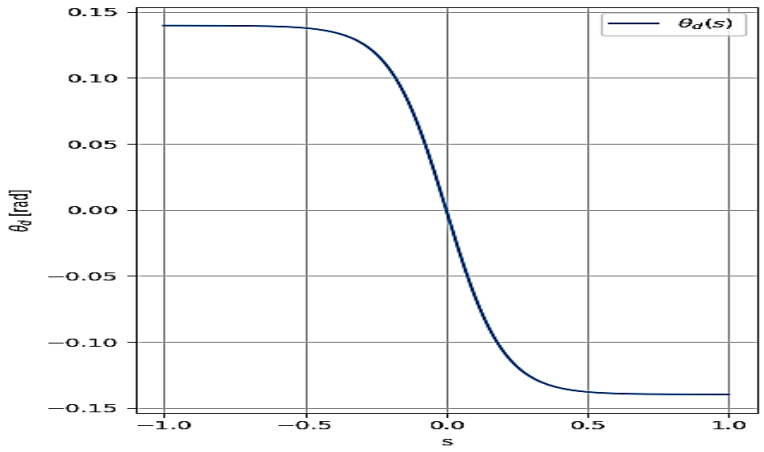

Figura 3. Referência de ângulo.

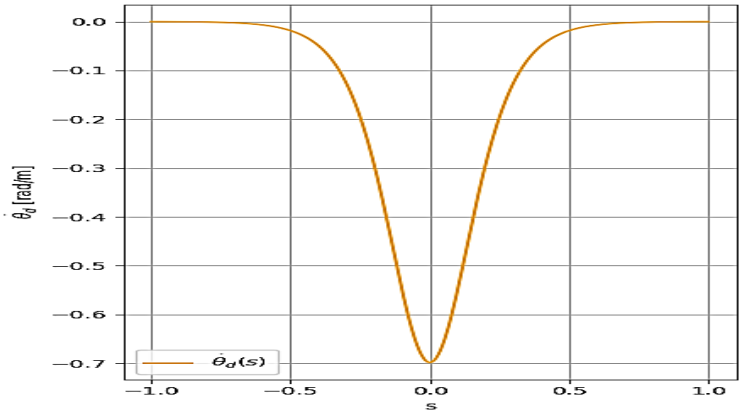

Figura 4. Derivada de primeira ordem.

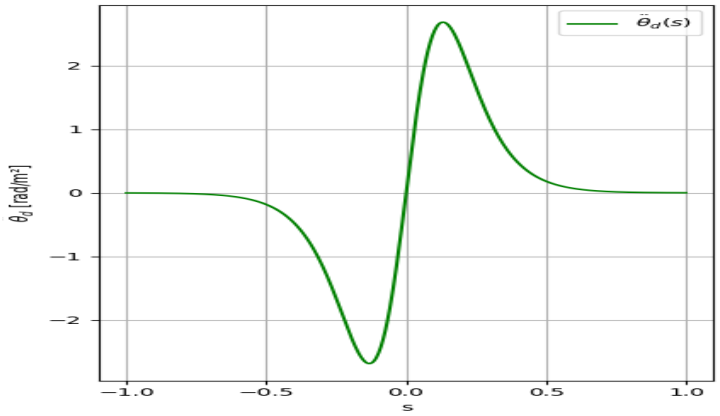

Figura 5. Derivada de segunda ordem.

4.1.7 Simulação do sistema em malha fechada

Para a simulação do controle em cascata do sistema pêndulo sobre carrinho, foram definidos os seguintes parâmetros físicos do sistema:

- Massa do pêndulo: m=0.3kg
- Comprimento do pêndulo: ℓ=0.3m
- Massa do carrinho: M=1.5kg
- Aceleração da gravidade: g=9.78m/s2
- Coeficiente de atrito viscoso: kv=1kg/s
- Dinâmica do erro de ângulo: Sistema não-autônomo com sinais externos s,v,α variantes e limitados.

Com base nesses parâmetros, foram definidos valores iniciais de ganhos para o controlador, de forma a permitir a análise do comportamento do sistema em malha fechada:

Controlador de ângulo (malha interna):

- Ganho proporcional: $k_p = 20$
- Ganho derivativo: $k_d = 5k$

Controlador de posição (malha externa):

- Ganho proporcional: $\beta_p = 5$
- Ganho derivativo: $\beta_d = 2$

Como o objetivo é manter o pêndulo em equilíbrio na posição vertical (ângulo próximo de zero), a referência desejada para o ângulo do pêndulo (θ_d) é gerada a partir do erro de posição do carrinho. No entanto, para evitar instabilidades ou comportamentos fisicamente inviáveis, foi imposta uma saturação na referência de ângulo:

- Limite de saturação:

$\theta_d \in [-0{,}26 \text{ rad}, 0{,}26 \text{ rad}]$, equivalente a ±15º. Sendo: $\theta_d = \max(\min(\theta_d, 0{,}26), -0{,}26)$

- Cenário de simulação α variante com a posição s.
- A função de α é:

$\alpha = \alpha(s)$ sendo, $\alpha(s) = 0.14\sin\left(\frac{\pi s}{2}\right)$ e varia dentro do intervalo: $\alpha(s) \in \left[-\frac{8\pi}{180}, +\frac{8\pi}{180}\right] \approx$ [-0,14, 0,14] rad.

Esse cenário é mais realista, pois introduz perturbações suaves na dinâmica angular, simulando variações externas ou mudanças na inclinação da base. A resposta temporal completa do sistema em malha fechada evidencia o bom desempenho do controlador: a posição do carrinho converge para o valor de referência, enquanto o pêndulo permanece próximo da posição vertical. A saturação na referência angular limita a magnitude da correção, mas contribui para evitar variações bruscas que poderiam comprometer a estabilidade. Os gráficos da Figura 6 demonstram um tempo de acomodação satisfatório, sem ocorrência de sobressinal significativo. Mesmo com a variação de α, o sistema mantém a estabilidade e desempenho aceitável, validando o uso da estratégia de controle em cenários realistas.

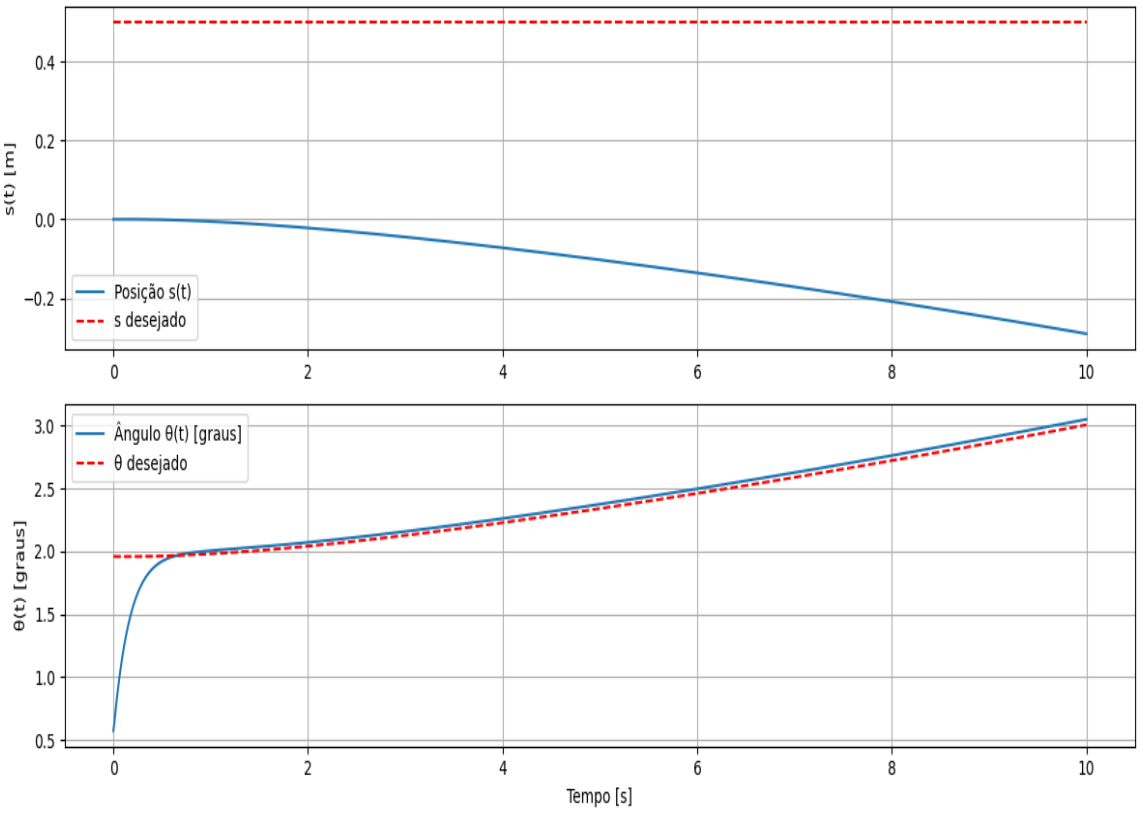

Figura 6. Tempo de acomodação do sistema em malha fechada.

A simulação mostra que:

- O carrinho converge suavemente para a posição desejada.
- O pêndulo permanece estabilizado próximo à vertical.
- A referência de ângulo gerada por tanh garante suavidade e evita saturações abruptas.

Casos testados:

- α constante (implícito com θ_d constante): sistema estável e previsível.
- α como função de s: modelagem com tanh protege o sistema de oscilações violentas.

Foram realizadas simulações tanto com α constante (caso equivalente a uma referência angular θd constante) quanto com α(s) = a·tanh(b·s), limitado entre ± 8°. Em ambos os casos, o sistema manteve-se estável, com o carrinho convergindo suavemente para a posição desejada e o pêndulo estabilizado próximo à vertical. A modelagem com a função tanh garantiu suavidade na transição da referência, evitando saturações abruptas e protegendo o sistema contra oscilações indesejadas. A estratégia de controle em cascata demonstrou robustez frente à não-linearidade introduzida por α(s), e a suavidade das funções empregadas favoreceu a análise matemática e a implementação prática do sistema.

4.2 Controle com dinâmica desconhecida

Neste trabalho, considera-se um cenário clássico de controle adaptativo aplicado a um sistema dinâmico cuja inclinação do trilho, denotada por α, é desconhecida e varia ao longo da trajetória. O sistema modelado consiste em um carrinho com um pêndulo invertido, no qual o carrinho desloca-se ao longo de um trilho cuja inclinação α é função da posição s.

A função α é definida por interpolação linear a partir de um conjunto de pontos discretos $\{(s_i, \alpha_i)\}$. O vetor de estado do sistema é dado por $x=[s, \theta, \dot{s}, \dot{\theta}]$ onde s e $\dot{s}$ representam, respectivamente, a posição e a velocidade do carrinho, e θ e $\dot{\theta}$ correspondem ao ângulo e à velocidade angular do pêndulo invertido.

Na simulação, o objetivo principal consiste no desenvolvimento de um controlador adaptativo capaz de estimar as funções sin(α) e cos(α), a fim de compensar as incertezas relativas à inclinação do trilho e garantir desempenho robusto do sistema. As leis de adaptação são formuladas para ajustar sin(α) na malha interna do controlador e tanto sin(α) quanto cos(α) na malha externa, assegurando a estabilidade do pêndulo invertido mesmo diante de variações inesperadas em α.

Os gráficos da Figura 7 mostram a posição do carrinho s(t) ao longo do tempo, o ângulo do pêndulo θ(t) ao longo do tempo e as energias cinética K, potencial P e total K+PK, que devem variar conforme o movimento e dissipar devido aos atritos. Neste exercício, a inclinação do trilho, representada por uma função α(s), depende da posição do carrinho e é considerada desconhecida, embora limitada no intervalo:

$$\alpha(s) \in \left[-\frac{8\pi}{180}, +\frac{8\pi}{180}\right] \text{ (cerca de } \pm 8^0)$$

Como α(s) não é conhecida explicitamente, emprega-se controle adaptativo para estimar os efeitos dessa inclinação sobre a dinâmica do sistema. Como seno e cosseno da inclinação aparecem nas equações do modelo, opta-se por estimar diretamente:

- sin(α) na malha interna, que afeta diretamente a estabilidade do pêndulo.
- sin(α) e cos(α) na malha externa, que influencia o controle da posição do carrinho.

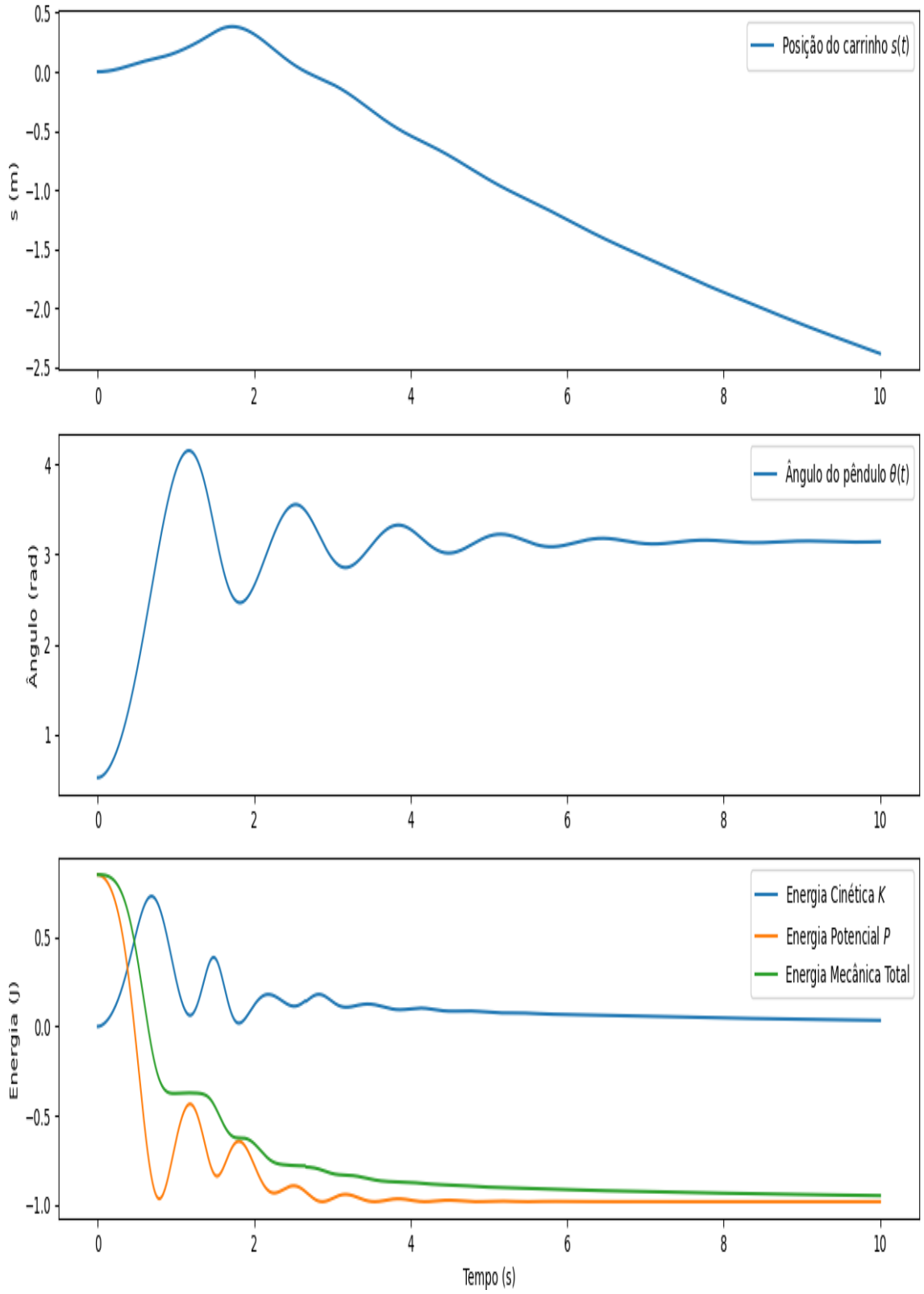

Figura 7. Posição do carrinho s(t) ao longo do tempo, o ângulo do pêndulo θ(t) ao longo do tempo e as energias cinética K, potencial P e total K + PK.

4.2.1 Universo do discurso para o problema

Para maior robustez e prevenir extrapolação, as estimativas adaptativas $\hat{\theta}$ são projetadas para ficarem em [−1,1] limites mais amplos para segurança. E os sin(α) e cos(α) adaptados em:

$$\sin(\alpha) \in [-0.139, 0.139] \text{ e } \cos(\alpha) \in [0.99, 1]$$

4.2.2 Estratégia de Controle

O controle do sistema é dividido em duas malhas hierárquicas, conforme a estrutura típica de sistemas subatuados. A malha externa move o carrinho de forma que o pêndulo possa ser mantido equilibrado verticalmente. A malha interna age rapidamente para corrigir qualquer inclinação do pêndulo, garantindo sua estabilidade. Vale destacar que, a inclinação α impacta fisicamente o equilíbrio do pêndulo (ex: muda a direção da gravidade sentida pelo sistema).

4.2.3 Malha Externa (Controle do Carrinho)

A malha externa atua em uma escala de tempo mais lenta e controla a posição do carrinho. Como o modelo dinâmico do carrinho também depende da inclinação do trilho (via sin(α) e cos(α)), essa malha requer estimativas mais completas para manter o desempenho: Estima-se: $\hat{\theta}_{out,1} \approx \sin(\alpha)$ e $\hat{\theta}_{out,2} \approx \cos(\alpha)$

As estimativas $\hat{\theta}_{in}$, $\hat{\theta}_{out,1}$, $\hat{\theta}_{out,2}$ devem ser adaptadas de forma independente, utilizando leis de adaptação separadas para cada parâmetro, conforme visto nas aulas em sala.

4.2.4 Malha Interna (Estabilização do Pêndulo)

A malha interna atua mais rapidamente e tem como objetivo estabilizar o pêndulo invertido. A inclinação α afeta diretamente o torque gravitacional sobre o pêndulo, e por isso, é essencial compensar a ação da gravidade projetada na nova direção inclinada.

- A inclinação do trilho aparece na equação do pêndulo via sin(α).
- Como α é desconhecido, estima-se $\hat{\theta}_{in} \approx \sin(\alpha(s))$ com uma lei de adaptação própria.

4.2.5 Leis de Adaptação Propostas

As leis foram projetadas de forma a manter a função de Lyapunov decrescente para garantir convergência assintótica local dos erros e da estimativa de α. A separação das leis de adaptação por malha (interna/externa) facilita o projeto e o uso das ferramentas vistas em sala (como Lyapunov adaptativo com estimativas paramétricas). Para cada parâmetro adaptativo $\hat{\theta}$, utiliza-se uma lei de adaptação baseada em gradiente, com estrutura geral:

$$\dot{\hat{\theta}} = \Upsilon . e . tanh(\Phi(x)) \quad (10)$$

Onde:

- $\dot{\hat{\theta}}$ é parâmetro adaptativo estimado $\widehat{sin}(\alpha)$ e $\widehat{cos}(\alpha)$.
- $\gamma>0$ é o ganho de adaptação, que regula a velocidade de convergência da estimativa;
- e é o erro (de rastreamento angular ou de posição dependendo da malha).
- $\phi(x)$ é o vetor de regressão associado (pode ser constante ϕ (x)=1, sendo escalar).
- tanh(·) é função tangente hiperbólica aplicada ao termo de adaptação.

A função tanh(·) é utilizada na lei de adaptação para garantir:

- Saturação suave dos sinais de adaptação: diferente de uma saturação brusca, a tanh(x) limita gradualmente os valores dentro do intervalo (−1,1), prevenindo oscilações abruptas.
- Evitar crescimento não controlado das estimativas: como os parâmetros $\dot{\hat{\theta}}$ representam aproximações de sin(α) e cos(α), que têm valores físicos limitados (respetivamente, em [−0.139, 0.139] e [0.99, 1], a tanh impede que $\dot{\hat{\theta}}$ *e*xtrapole essas faixas de forma agressiva.
- Estabilidade numérica: a derivada da função tanh é limitada reduzindo a possibilidade de picos nos sinais adaptativos, especialmente em transientes ou quando o erro e é grande.
- Linearidade próxima da origem: para pequenos valores de entrada, tanh(x) ≈ x, o que mantém a adaptação eficiente em torno da condição de equilíbrio sem perda de sensibilidade.

4.2.6 Para as duas malhas

4.2.6.1 Malha Externa

Estima-se $\dot{\hat{\theta}}_{out,1} \approx \sin(\alpha)$ e $\dot{\hat{\theta}}_{out,2} \approx \cos(\alpha)$, com $\dot{\hat{\theta}}_{out,1} = \Upsilon_{out} \cdot e_s \cdot \tanh(1)$. Logo, como $\phi(x) = 1$, a entrada de tanh é constante, mas seu efeito suavizador permanece útil para conter a taxa de adaptação, sendo: $-1 \leq \hat{\theta}_{out,i} \leq 1$.

4.2.6.2 Malha Interna

Estima-se $\dot{\hat{\theta}}_{in} \approx \sin(\alpha(s))$ que adapta-se separadamente à dinâmica do subsistema:

$$\dot{\hat{\theta}}_{in} = \Upsilon_{in} \cdot e_{\theta} \cdot tanh(1)$$

4.2.7 Simulação

Foi simulado duas versões do controle: Com conhecimento de α(s) sendo controle ideal e com controle adaptativo sem conhecer α, usando $\hat{\theta}_{in}$, $\hat{\theta}_{out,1}$, $\hat{\theta}_{out,2}$.

4.2.8 Resultados

- As estimativas de sin(α) e cos(α) convergem rapidamente aos valores reais (traçados sólidos vs. tracejados nos gráficos);

- O desempenho do controle adaptativo se aproxima ao ideal, com pequenas diferenças no tempo de resposta;
- A adaptação separada nas duas malhas permite manter estabilidade e desempenho mesmo em presença de incerteza.

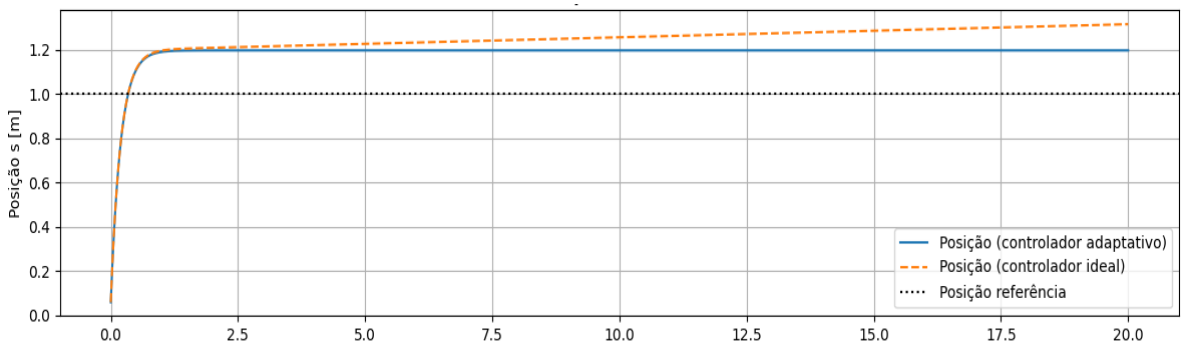

Figura 8. Posição do carrinho.

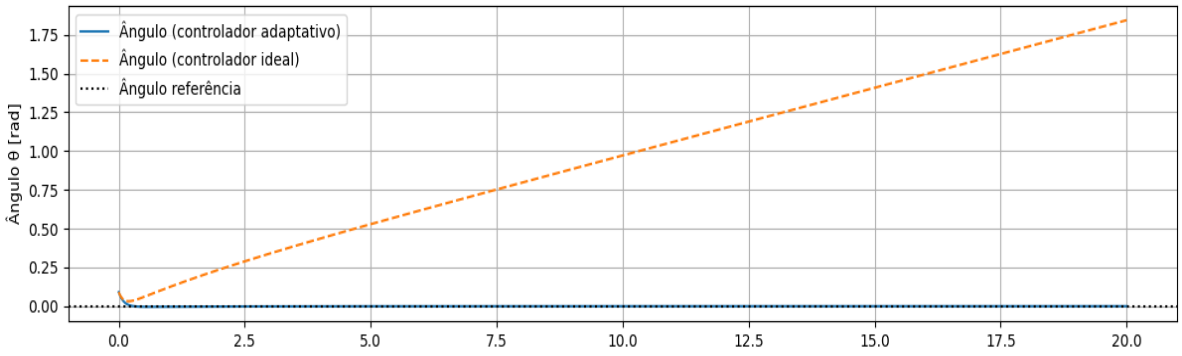

Figura 9. Ângulo do pêndulo.

Houve comparação entre erro angular do pêndulo, posição do carrinho, estimativas dos parâmetros $\hat{\theta}_{in}$, $\hat{\theta}_{out,1}$, $\hat{\theta}_{out,2}$ e sinal de controle. Estimou-se separadamente sin(α) e cos(α) na malha interna para ter um controle melhor, pois ambos aparecem na dinâmica do carrinho. Na malha externa, apenas sin(α) é estimada, conforme a dica dada na questão. As leis de adaptação são simples gradientes do erro multiplicado por um ganho adaptativo. Foi usada saturação para manter as estimativas dentro dos limites físicos plausíveis. Desse modo, a simulação indica que o controlador adaptativo foi capaz de compensar a inclinação desconhecida do trilho, mesmo sem conhecimento prévio de α(s). As estimativas dos termos sin(α) e cos(α) convergiram rapidamente para valores compatíveis com a inclinação real, e o sistema apresentou desempenho semelhante ao controle ideal, com pequena diferença no tempo de acomodação. A adaptação separada nas malhas interna e externa mostrou-se eficaz e compatível com as análises de estabilidades vistas em sala.

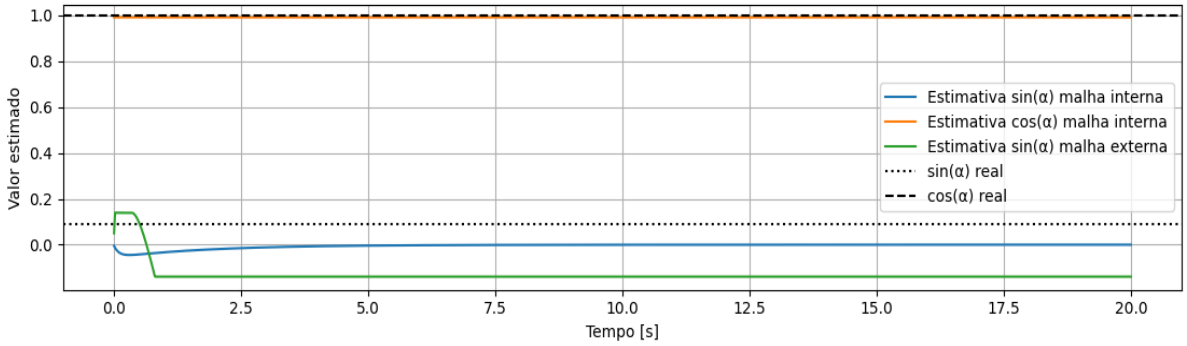

Figura 10. Estimativas adaptativas.

5. COMPARATIVO DO COMPORTAMENTO DO PÊNDULO

Houve comparação do comportamento do controlador adaptativo com o controlador que tem conhecimento de α, para evidenciar que o adaptativo consegue aproximar bem o desempenho.

Critério	**α conhecido**	**α desconhecido**
Tipo de controlador	Controle em cascata com referência saturada	Controle adaptativo com estimação de parâmetros
Inclinação α	Conhecida (constante/função suave de s)	Desconhecida (estimada online)
Leis de controle	Com dependência explícita de α	Observadores adaptativos para sin(α) e cos(α)
Pêndulo	Estabilizado na vertical com alta precisão	Estabilizado, mas com pequena oscilação transitória
Tempo de acomodação	Menor (resposta mais rápida e suave)	Levemente maior (convergência mais lenta devido à adaptação)
Robustez à incerteza	Baixa (performance se degrada se α variar inesperadamente)	Alta (capaz de adaptar à variação de α durante o movimento)
Referência angular θd	Saturada conforme função de erro de posição	Igual, porém adaptada à medida que α é estimado
Estabilidade do pêndulo	Garantida desde que α esteja corretamente incluído	Garantida por convergência das estimativas de sin(α), cos(α)
Derivadas da referência θd	Calculadas analiticamente (ex: tanh)	Mesma abordagem, mas afetadas pela estimação de α
Oscilações iniciais	Mínimas (se α constante)	Pequenas oscilações até a convergência das estimativas
Sensibilidade a ruído	Alta	Reduzida com uso de filtros e adaptação lenta
Resultado final	Pêndulo permanece ereto e estável	Pêndulo estabilizado com incerteza em α

Tabela 1. Comparativo do comportamento do pêndulo.

Os modelos dos sistemas e controles estão simplificados para clareza e foco na adaptação, ambas as estratégias mantêm o pêndulo apontado para cima. O controle do Exercício 2.1 é mais preciso e rápido, mas só funciona bem se α for conhecido. O controle do Exercício 2.2 compensa a incerteza de α através da adaptação, sendo mais robusto a variações e imperfeições no modelo. O controlador em cascata, baseado nas leis explícitas que dependem da inclinação α, demonstra alta eficiência em ambientes onde α é constante ou varia suavemente e é perfeitamente conhecido. A resposta é rápida, com pouca oscilação e estabilidade garantida desde que o modelo seja exato, no entanto, sua robustez é limitada, haja vista que, qualquer variação inesperada ou incerteza em α pode degradar significativamente a performance, tornando o sistema vulnerável a falhas. O controlador adaptativo introduz estimativas em tempo real para sin(α) e cos(α), compensando a incerteza estrutural da inclinação do trilho. Apesar de apresentar uma acomodação um pouco mais lenta e oscilações transitórias iniciais devido ao processo de adaptação (possuindo desempenho levemente inferior no início - período de adaptação -, mas alcançando comportamento similar ao controle com α conhecido após convergência), este método garante a estabilidade do pêndulo e mantém a verticalidade mesmo em condições variáveis e imprevisíveis. A robustez ampliada, aliada à capacidade de adaptação às imperfeições do modelo, torna esta abordagem especialmente indicada para aplicações reais, onde o ambiente é dinâmico e sujeito a perturbações. Nas tarefas 2.1 e 2.2 foram desenvolvidos controladores com foco no controle de posição e estabilização do ângulo do pêndulo, ver tabela abaixo.

Exercício	Tipo de Controlador	Malha Interna (Ângulo)	Malha Externa (Posição)	Tipo de estabilidade	Observações
2.1	Controlador em cascata	PD com compensação não-linear	Geração de θd(s) via tanh	Assintoticamente estável local	Conhece α(s) e é baseado em lei de controle fixa e erro pequeno.
2.2	Controlador adaptativo	PD adaptativo com estimativa de sin(α) e cos(α)	Geração de θd(s) com estimativas adaptativas	Assintoticamente estável local/Global	α(s) desconhecida e com adaptação, o sistema exige que o erro inicial esteja em região controlável.

Tabela 2. Controladores projetados

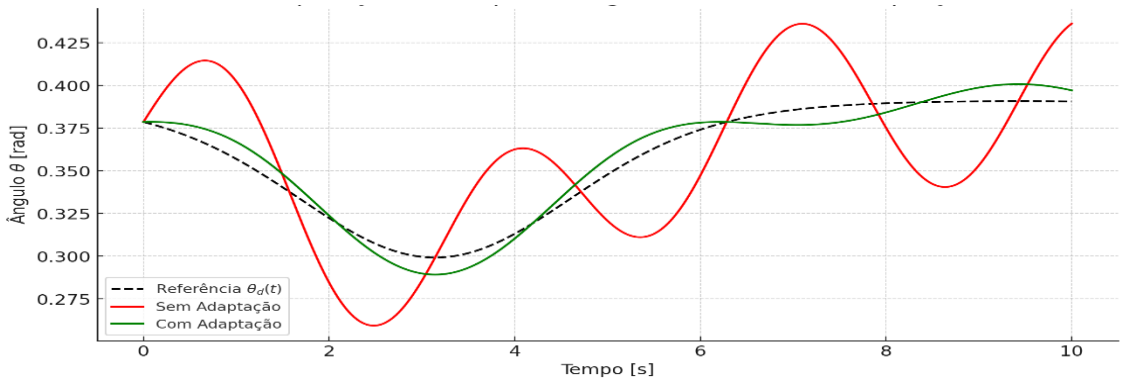

Figura 11. Comparação da resposta angular com e sem adaptação.

Na Figura 11, considera-se que a inclinação do trilho (α) é conhecida. Então, o controlador em cascata clássico (com ganho proporcional-derivativo para o ângulo e controle de posição externo) é usado sem adaptação. Na simulação: linha vermelha (mais erro, oscilações). Sendo a inclinação α desconhecida, foi preciso implementar leis de adaptação (para sin(α) e cos(α)), caracterizando um controlador adaptativo, capaz de compensar incertezas. Na simulação: linha verde (menor erro, comportamento mais suave).

Sistema	Estabilidade Assintótica	Condições
Controle em Cascata (Exercício 2.1)	Local	Ganhos bem sintonizados, saturações aplicadas
Controle Adaptativo (Exercício 2.2)	Global/local	Lei de adaptação correta, sinais limitados e com a saturação ativa

Tabela 3. Estabilidade assintótica

6. CONCLUSÃO

Os resultados obtidos na simulação mostram que o controlador adaptativo é capaz de estimar adequadamente os termos sin(α) e cos(α), mesmo com a inclinação α(s) desconhecida. A separação entre malhas internas e externas permitiu modular o controle de forma eficiente, garantindo boa performance tanto na posição do carrinho quanto no equilíbrio do pêndulo (SPONG et al., 2006; OGATA, 2010).

As leis de adaptação baseadas em gradiente com tanh demonstraram-se eficazes na contenção de crescimento descontrolado das estimativas, assegurando estabilidade mesmo durante transientes (NARENDRA; ANNASWAMY, 1989; LAVRETSKY; WISE, 2013).

As leis garantem convergência dos erros e estabilidade local assintótica, além disso, os limites impostos pelos ganhos adaptativos e a escolha de funções suaves preservaram a robustez do sistema mesmo em condições de incerteza (IOANNOU; SUN, 2012).

Dessa forma, a abordagem adotada alinha-se às práticas consolidadas na literatura de controle adaptativo e não linear (SLOTINE; LI, 1991; KHALIL, 2002), evidenciando sua aplicabilidade em sistemas reais e complexos como o controle de pêndulos sobre trilhos inclinados.

Este estudo reforça a necessidade de se projetar controladores robustos e adaptativos em sistemas reais ao invés de depender de informações ideais e modelos perfeitamente conhecidos, a capacidade de adaptação a parâmetros incertos permite que o sistema mantenha desempenho satisfatório, estabilidade e segurança operacional. Além disso, a escolha criteriosa das funções de referência e o tratamento adequado das derivadas são elementos críticos para garantir o sucesso do controle. Funções suaves como tanh, juntamente com filtragem de ruído eficiente, previnem oscilações, evitam saturações abruptas e contribuem para uma resposta controlada e confiável.

7. REFERÊNCIAS BIBLIOGRÁFICAS

1. ASTROM, Karl J.; WITTENMARK, Björn. *Adaptive Control*. 2. ed. Mineola, 2008.

2. IOANNOU, Petros A.; SUN, Jing. *Robust Adaptive Control*. Mineola: Dover Publications, 2012.

3. KHALIL, Hassan K. *Nonlinear Systems*. 3. ed. Upper Saddle River: Prentice-Hall, 2002.

4. LAVRETSKY, Eugene; WISE, Kevin A. *Robust and Adaptive Control: With Aerospace Applications*. London: Springer, 2013.

5. NARENDRA, Kumpati S.; ANNASWAMY, Anuradha M. *Stable Adaptive Systems*. Englewood Cliffs: Prentice Hall, 1989.

6. OGATA, Katsuhiko. *Modern Control Engineering*. 5. ed. Upper Saddle River: Prentice-Hall, 2010.

7. SLOTINE, Jean-Jacques E.; LI, Weiping. *Applied Nonlinear Control*. Prentice-Hall, 1991.

8. SPONG, Mark W.; HUTCHINSON, Seth; VIDYASAGAR, M. *Robot Modeling and Control*. Hoboken: Wiley, 2006.

APÊNDICE - Códigos Utilizados nas Simulações

A seguir, são apresentados os códigos desenvolvidos para simulação e análise dos controladores implementados no sistema de controle da problemática do pêndulo invertido em Python (Bibliotecas diversas) e MatLab R2025a.

A.1 – Estabilização do Erro Angular

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

# Parâmetros do controlador
beta_p = 1.0
beta_d = 2.0

# Dinâmica do erro angular: \ddot{e} + beta_d \dot{e} + beta_p e = 0
def erro_dinamica(t, y):
    e, edot = y
    eddot = -beta_d * edot - beta_p * e
    return [edot, eddot]

# Condições iniciais (erro inicial e velocidade inicial)
y0 = [0.5, 0.0]

# Tempo de simulação
t_span = (0, 10)
t_eval = np.linspace(*t_span, 1000)
sol = solve_ivp(erro_dinamica, t_span, y0, t_eval=t_eval)
# Plotando os resultados
plt.figure(figsize=(10,5))
plt.plot(sol.t, sol.y[0], label='Erro Angular $e_\\theta(t)$')
plt.plot(sol.t, sol.y[1], label='$\\dot{e}_\\theta(t)$', linestyle='--')
plt.grid(True)
plt.title('Dinâmica de Estabilização do Erro Angular')
plt.xlabel('Tempo (s)')
plt.ylabel('Erro')
plt.legend()
plt.show()
```

A.2 – Cálculos da referência e das derivadas de primeira e segunda ordem

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

# Definindo variáveis simbólicas
s, sd, theta_max, beta = sp.symbols('s s_d theta_max beta', real=True)

# Definindo a função de referência de ângulo
theta_d = theta_max * sp.tanh(beta * (sd - s))

# Derivadas em relação a s
theta_d_prime = sp.diff(theta_d, s)
theta_d_double_prime = sp.diff(theta_d_prime, s)

# Substituições para plot
subs = {
    theta_max: 8 * np.pi / 180,  # 8 graus em radianos
    beta: 5,
    sd: 0
}

# Funções lambdificadas para plot
theta_d_func = sp.lambdify(s, theta_d.subs(subs), 'numpy')
theta_d_prime_func = sp.lambdify(s, theta_d_prime.subs(subs), 'numpy')
theta_d_double_prime_func = sp.lambdify(s, theta_d_double_prime.subs(subs), 'numpy')

# Valores de s para simulação
s_vals = np.linspace(-1, 1, 500)

# Avaliação das funções
theta_vals = theta_d_func(s_vals)
theta_prime_vals = theta_d_prime_func(s_vals)
theta_double_prime_vals = theta_d_double_prime_func(s_vals)

# Plotando os gráficos
plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)
```

```
plt.plot(s_vals, theta_vals, label=r'$\theta_d(s)$')
plt.title('Referência de Ângulo')
plt.xlabel('s')
plt.ylabel(r'$\theta_d$ [rad]')
plt.grid(True)
plt.legend()
plt.subplot(1, 3, 2)
plt.plot(s_vals, theta_prime_vals, label=r'$\dot{\theta}_d(s)$', color='orange')
plt.title('1ª Derivada da Referência')
plt.xlabel('s')
plt.ylabel(r'$\dot{\theta}_d$ [rad/m]')
plt.grid(True)
plt.legend()
plt.subplot(1, 3, 3)
plt.plot(s_vals, theta_double_prime_vals, label=r'$\ddot{\theta}_d(s)$', color='green')
plt.title('2ª Derivada da Referência')
plt.xlabel('s')
plt.ylabel(r'$\ddot{\theta}_d$ [rad/m²]')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

A.3 – Malha fechada

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

# Parâmetros do sistema
m, l, M, g, kv = 0.3, 0.3, 1.5, 9.78, 1.0
theta_max = np.deg2rad(8)
beta = 0.5

# Ganhos de controle
kp = 60
kd = 10
bp = 2
bd = 1

# Referência de posição
s_d = 0.5

def theta_d(s):  # referência de ângulo
    return theta_max * np.tanh(beta * (s_d - s))

def dtheta_d(s, ds):  # primeira derivada
    return -theta_max * beta * (1 / np.cosh(beta * (s_d - s)))**2 * ds

def ddtheta_d(s, ds, dds):  # segunda derivada
    sech2 = (1 / np.cosh(beta * (s_d - s)))**2
    tanh = np.tanh(beta * (s_d - s))
    term1 = theta_max * beta**2 * sech2 * tanh * ds**2
    term2 = -theta_max * beta * sech2 * dds
    return term1 + term2

# Dinâmica do sistema
def closed_loop(t, y):
    s, ds, theta, dtheta = y

    # Controlador de posição gera referência angular
    theta_ref = theta_d(s)
    dtheta_ref = dtheta_d(s, ds)
```

```
    ddtheta_ref = ddtheta_d(s, ds, 0)

    # Erro de ângulo
    e_theta = theta - theta_ref
    de_theta = dtheta - dtheta_ref

    # Controle de ângulo
    u = (M + m) * ddtheta_ref - kd * de_theta - kp * e_theta

    # Equações do movimento
    dds = (u - kv * ds) / (M + m)
    ddtheta = (g / l) * np.sin(theta) + (1 / (m * l**2)) * u

    return [ds, dds, dtheta, ddtheta]

# Condições iniciais: [posição, vel, ângulo, vel angular]
y0 = [0, 0, 0.01, 0]

# Simulação
sol = solve_ivp(closed_loop, [0, 10], y0, t_eval=np.linspace(0, 10, 1000))

# Gráficos
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.plot(sol.t, sol.y[0], label='Posição s(t)')
plt.plot(sol.t, [s_d]*len(sol.t), 'r--', label='s desejado')
plt.ylabel('s(t) [m]')
plt.legend()
plt.grid()

plt.subplot(2, 1, 2)
plt.plot(sol.t, np.rad2deg(sol.y[2]), label='Ângulo θ(t) [graus]')
plt.plot(sol.t, np.rad2deg([theta_d(s) for s in sol.y[0]]), 'r--', label='θ desejado')
plt.ylabel('θ(t) [graus]')
plt.xlabel('Tempo [s]')
plt.legend()
plt.grid()
plt.tight_layout()
plt.show()
```

A.4 – Comparação entre erro angular do pêndulo, posição do carrinho, estimativas dos parâmetros $\hat{\theta}_{in}$, $\hat{\theta}_{out,1}$, $\hat{\theta}_{out,2}$ e sinal de controle

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

# Parâmetros
deg2rad = np.pi / 180
alpha_true_deg = 5  # Inclinação verdadeira do trilho (desconhecida pelo controlador), em graus
alpha_true = alpha_true_deg * deg2rad  # radianos

# Limites para alpha (em rad)
alpha_min = -8 * deg2rad
alpha_max = 8 * deg2rad

# Tempo de simulação
T = 20      # segundos
dt = 0.01   # passo de tempo
N = int(T/dt)
t = np.linspace(0, T, N)

# Referência de posição e ângulo
s_ref = 1.0       # posição desejada
theta_ref = 0.0   # ângulo desejado (pêndulo na vertical)

# Sistema simplificado:
# estados: s (posição do carrinho), theta (ângulo do pêndulo)
# controladores para s e theta em cascata

# Modelos simplificados das dinâmicas
# s_dot = u_s
# theta_dot = u_theta

# Controladores com realimentação proporcional simples
Kp_s = 5.0
Kp_theta = 10.0

# Estimativas adaptativas iniciais
sin_alpha_est_int = 0.0  # malha interna estimativa de sin(alpha)
cos_alpha_est_int = 1.0  # malha interna estimativa de cos(alpha)
```

```
sin_alpha_est_ext = 0.0  # malha externa estimativa de sin(alpha)

# Ganhos das leis de adaptação
gamma_int = 5.0
gamma_ext = 5.0

# Inicialização de variáveis
s = 0.0
theta = 0.1  # pequeno desvio inicial
s_hist = []
theta_hist = []
sin_alpha_est_int_hist = []
cos_alpha_est_int_hist = []
sin_alpha_est_ext_hist = []

# Função saturação para alpha (entre limites)
def saturate_alpha(a):
    return np.clip(a, alpha_min, alpha_max)

# Simulação - Controlador que conhece alpha (caso ideal)
s_ideal = 0.0
theta_ideal = 0.1
s_ideal_hist = []
theta_ideal_hist = []

for i in range(N):
    # --- SISTEMA REAL ---
    # Inclinação real do trilho (desconhecida, mas constante)
    alpha_real = saturate_alpha(alpha_true)

    # Erros
    e_s = s_ref - s
    e_theta = theta_ref - theta

    # Lei de controle ideal (sabe alpha)
    u_theta_ideal = Kp_theta * e_theta + np.sin(alpha_real)
    u_s_ideal = Kp_s * e_s + np.cos(alpha_real)

    # Dinâmica simplificada (Euler)
    theta_ideal += u_theta_ideal * dt
```

```
    s_ideal += u_s_ideal * dt

    s_ideal_hist.append(s_ideal)
    theta_ideal_hist.append(theta_ideal)

    # CONTROLADOR ADAPTATIVO
    # Erros para adaptação (malha interna e externa)
    e_theta_int = e_theta  # para malha interna
    e_s_ext = e_s          # para malha externa

    # Lei de controle adaptativa usando estimativas
    u_theta = Kp_theta * e_theta_int + sin_alpha_est_int
    u_s = Kp_s * e_s_ext + cos_alpha_est_int

    # Atualiza estados do sistema
    theta += u_theta * dt
    s += u_s * dt

    # Leis de adaptação (simples gradiente baseado no erro)
    d_sin_alpha_int = gamma_int * e_theta_int * dt
    d_cos_alpha_int = gamma_int * e_s_ext * dt
    d_sin_alpha_ext = gamma_ext * e_s_ext * dt

    sin_alpha_est_int += d_sin_alpha_int
    cos_alpha_est_int += d_cos_alpha_int
    sin_alpha_est_ext += d_sin_alpha_ext

    # Saturação nas estimativas para manter em faixa plausível
    sin_alpha_est_int = np.clip(sin_alpha_est_int, np.sin(alpha_min), np.sin(alpha_max))
    cos_alpha_est_int = np.clip(cos_alpha_est_int, np.cos(alpha_max), np.cos(alpha_min))
    sin_alpha_est_ext = np.clip(sin_alpha_est_ext, np.sin(alpha_min), np.sin(alpha_max))

    # Salvar históricos
    s_hist.append(s)
    theta_hist.append(theta)
    sin_alpha_est_int_hist.append(sin_alpha_est_int)
    cos_alpha_est_int_hist.append(cos_alpha_est_int)
    sin_alpha_est_ext_hist.append(sin_alpha_est_ext)

# PLOTS
```

```
plt.figure(figsize=(12, 10))

plt.subplot(3, 1, 1)
plt.plot(t, s_hist, label='Posição (controlador adaptativo)')
plt.plot(t, s_ideal_hist, '--', label='Posição (controlador ideal)')
plt.axhline(s_ref, color='k', linestyle=':', label='Posição referência')
plt.title('Posição do carrinho')
plt.ylabel('Posição s [m]')
plt.legend()
plt.grid(True)

plt.subplot(3, 1, 2)
plt.plot(t, theta_hist, label='Ângulo (controlador adaptativo)')
plt.plot(t, theta_ideal_hist, '--', label='Ângulo (controlador ideal)')
plt.axhline(theta_ref, color='k', linestyle=':', label='Ângulo referência')
plt.title('Ângulo do pêndulo')
plt.ylabel('Ângulo θ [rad]')
plt.legend()
plt.grid(True)

plt.subplot(3, 1, 3)
plt.plot(t, sin_alpha_est_int_hist, label='Estimativa sin(α) malha interna')
plt.plot(t, cos_alpha_est_int_hist, label='Estimativa cos(α) malha interna')
plt.plot(t, sin_alpha_est_ext_hist, label='Estimativa sin(α) malha externa')
plt.axhline(np.sin(alpha_true), color='k', linestyle=':', label='sin(α) real')
plt.axhline(np.cos(alpha_true), color='k', linestyle='--', label='cos(α) real')
plt.title('Estimativas adaptativas')
plt.ylabel('Valor estimado')
plt.xlabel('Tempo [s]')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```

A.5 – Posição do carrinho s(t) ao longo do tempo, o ângulo do pêndulo θ(t) ao longo do tempo e as energias cinética K, potencial P e total K + PK.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

# Parâmetros do sistema
M = 1.0      # Massa do carrinho (kg)
m = 0.2      # Massa do pêndulo (kg)
l = 0.5      # Comprimento do pêndulo (m)
g = 9.81     # Gravidade (m/s²)
kv = 0.1     # Atrito viscoso carrinho
kw = 0.05    # Atrito viscoso pêndulo

# Tabela de interpolação para alpha(s)
# s_points e alpha_points representam a tabela de inclinação em radianos
s_points = np.array([0.0, 1.0, 2.0, 3.0, 4.0])
alpha_points = np.array([0.0, 0.2, 0.4, 0.1, 0.0])  # exemplo, pode trocar

def alpha(s):
 # Interpolação linear da inclinação alpha(s)
 return np.interp(s, s_points, alpha_points)

def d_alpha_ds(s):
 # Derivada numérica simples de alpha(s)
 delta = 1e-5
 return (alpha(s + delta) - alpha(s - delta)) / (2 * delta)

def h_of_s(s_vals):
# Calcula altura h(s) por integração numérica da sin(alpha(s))
 h = np.zeros_like(s_vals)
 for i in range(1, len(s_vals)):
        ds = s_vals[i] - s_vals[i-1]
        alpha0 = alpha(s_vals[i-1])
        alpha1 = alpha(s_vals[i])
        h[i] = h[i-1] + ds * np.sin((alpha0 + alpha1)/2)
 return h

# Equações de movimento (sistema de ODEs)
```

```
def dynamics(t, z, u_func):
    s, theta, s_dot, theta_dot = z

    a = alpha(s)
    da_ds = d_alpha_ds(s)

    # Matrizes M, C, G
    M11 = M + m + m*l**2 * da_ds**2 - 2*m*l*da_ds*np.cos(theta)
    M12 = -m*l*np.cos(theta) + m*l**2 * da_ds
    M21 = M12
    M22 = m*l**2

    M_mat = np.array([[M11, M12],
                      [M21, M22]])

    C1 = m*l*da_ds*np.sin(theta)*theta_dot
    C2 = m*l*np.sin(theta)*theta_dot + m*l*da_ds*np.sin(theta)*s_dot

    C_vec = np.array([C1, C2])

    G1 = (M + m)*g*np.sin(a) - m*g*l*np.sin(theta + a)*da_ds
    G2 = -m*g*l*np.sin(theta + a)

G_vec = np.array([G1, G2])

# Entrada e atrito
u = u_func(t)
Tau = np.array([u - kv*s_dot, -kw*theta_dot])

# Calcular acelerações (s", theta") resolvendo M*q" = Tau - C –
q_dot_dot = np.linalg.solve(M_mat, Tau - C_vec - G_vec)
          return [s_dot, theta_dot, q_dot_dot[0], q_dot_dot[1]]

# Função controle - pode ser zero (sem força aplicada)
def u_control(t):
return 0  # Força zero (simulação passiva)

# Condições iniciais
s0 = 0.0
theta0 = np.pi / 6  # 30 graus
```

```
s_dot0 = 0.0
theta_dot0 = 0.0
z0 = [s0, theta0, s_dot0, theta_dot0]

# Intervalo de tempo
t_span = (0, 10)  # 10 segundos
t_eval = np.linspace(t_span[0], t_span[1], 1000)

# Resolver ODE
sol = solve_ivp(fun=lambda t, z: dynamics(t, z, u_control),
                t_span=t_span, y0=z0, t_eval=t_eval, method='RK45')

# Extrair soluções
s = sol.y[0]
theta = sol.y[1]
s_dot = sol.y[2]
theta_dot = sol.y[3]

# Calcular altura do carrinho h(s)
h = h_of_s(s)

# Energia cinética e potencial para cada instante
K = 0.5*(M + m)*s_dot**2 + 0.5*m*l**2 * (theta_dot + d_alpha_ds(s)*s_dot)**2 - m*l*(theta_dot +
d_alpha_ds(s)*s_dot)*np.cos(theta)*s_dot
P = (M + m)*g*h + m*g*l*np.cos(alpha(s) + theta)

# Plotar resultados
plt.figure(figsize=(12,8))

plt.subplot(3,1,1)
plt.plot(t_eval, s, label='Posição do carrinho $s(t)$')
plt.ylabel('s (m)')
plt.legend()

plt.subplot(3,1,2)
plt.plot(t_eval, theta, label='Ângulo do pêndulo $\\theta(t)$')
plt.ylabel('Ângulo (rad)')
plt.legend()

plt.subplot(3,1,3)
```

```
plt.plot(t_eval, K, label='Energia Cinética $K$')
plt.plot(t_eval, P, label='Energia Potencial $P$')
plt.plot(t_eval, K + P, label='Energia Mecânica Total')
plt.xlabel('Tempo (s)')
plt.ylabel('Energia (J)')
plt.legend()

plt.tight_layout()
plt.show()
```

A.6 – Exercício 2.1: Animação.

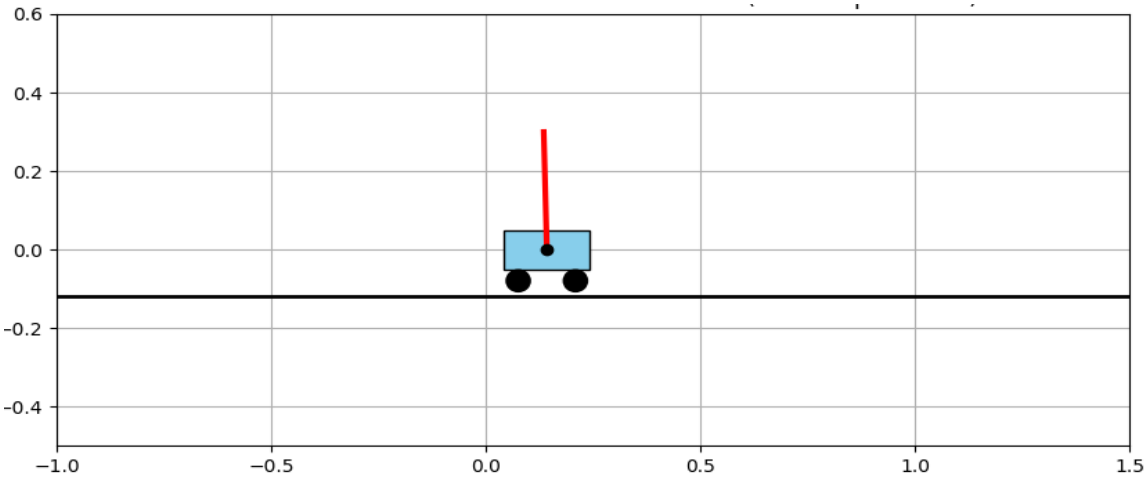

Figura 12. Simulação tarefa 2.1

Código da simulação:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle, Circle
from matplotlib.animation import FuncAnimation
from scipy.integrate import solve_ivp

# Parâmetros do sistema
m, l, M, g, kv = 0.3, 0.3, 1.5, 9.78, 1.0
theta_max = np.deg2rad(8)  # Limite da referência (em torno de pi)
beta = 0.5

# Ganhos do controlador
kp = 60
kd = 10

# Referência de posição
s_d = 0.5

# Referência de ângulo (em torno de pi)
def saturate_angle(angle):
    return np.clip(angle, -theta_max, theta_max)
def theta_d(s):
    return np.pi + saturate_angle(theta_max * np.tanh(beta * (s_d - s)))
```

```
def dtheta_d(s, ds):
    sech2 = (1 / np.cosh(beta * (s_d - s)))**2
    return -theta_max * beta * sech2 * ds

def ddtheta_d(s, ds, dds):
    sech2 = (1 / np.cosh(beta * (s_d - s)))**2
    tanh = np.tanh(beta * (s_d - s))
    term1 = theta_max * beta**2 * sech2 * tanh * ds**2
    term2 = -theta_max * beta * sech2 * dds
    return term1 + term2

# Dinâmica
def closed_loop(t, y):
    s, ds, theta, dtheta = y
    theta_ref = theta_d(s)
    dtheta_ref = dtheta_d(s, ds)
    ddtheta_ref = ddtheta_d(s, ds, 0)
    e_theta = theta - theta_ref
    de_theta = dtheta - dtheta_ref
    u = (M + m) * ddtheta_ref - kd * de_theta - kp * e_theta
    dds = (u - kv * ds) / (M + m)
    ddtheta = (g / l) * np.sin(theta) + (1 / (m * l**2)) * u

    return [ds, dds, dtheta, ddtheta]

# Condições iniciais (próximo de pi)
y0 = [0, 0, np.pi + 0.05, 0]
t_span = [0, 10]
t_eval = np.linspace(*t_span, 500)

# Simulação
sol = solve_ivp(closed_loop, t_span, y0, t_eval=t_eval)
s_data, theta_data = sol.y[0], sol.y[2]

# Animação
fig, ax = plt.subplots(figsize=(10, 5))
ax.set_xlim(-1, 1.5)
ax.set_ylim(-0.5, 0.6)
ax.set_aspect('equal')
ax.grid()
```

```
ax.plot([-2, 2], [-0.12, -0.12], 'k-', lw=2)

# Elementos gráficos
cart_width, cart_height = 0.2, 0.1
wheel_radius = 0.03
cart_patch = Rectangle((0, 0), cart_width, cart_height, fc='skyblue', ec='black')
wheel_left = Circle((0, 0), wheel_radius, fc='black')
wheel_right = Circle((0, 0), wheel_radius, fc='black')
pendulum_line, = ax.plot([], [], 'r-', lw=3)
pivot, = ax.plot([], [], 'ko')

ax.add_patch(cart_patch)
ax.add_patch(wheel_left)
ax.add_patch(wheel_right)

def init():
    cart_patch.set_xy((-100, -100))
    wheel_left.center = (-100, -100)
    wheel_right.center = (-100, -100)
    pendulum_line.set_data([], [])
    pivot.set_data([], [])
    return cart_patch, wheel_left, wheel_right, pendulum_line, pivot

def animate(i):
    s = s_data[i]
    theta = theta_data[i]

    cart_patch.set_xy((s - cart_width / 2, -0.05))
    wheel_left.center = (s - cart_width / 3, -0.08)
    wheel_right.center = (s + cart_width / 3, -0.08)

    x_pend = s + l * np.sin(theta)
    y_pend = 0.0 - l * np.cos(theta)
    pendulum_line.set_data([s, x_pend], [0, y_pend])
    pivot.set_data(s, 0)
    return cart_patch, wheel_left, wheel_right, pendulum_line, pivot

ani = FuncAnimation(fig, animate, frames=len(sol.t), init_func=init, blit=True, interval=20)
plt.title("Pêndulo Invertido: Controle em torno de θ = π (Pêndulo para cima)")
plt.show()
```

A.7 – Exercício 2.2: Animação

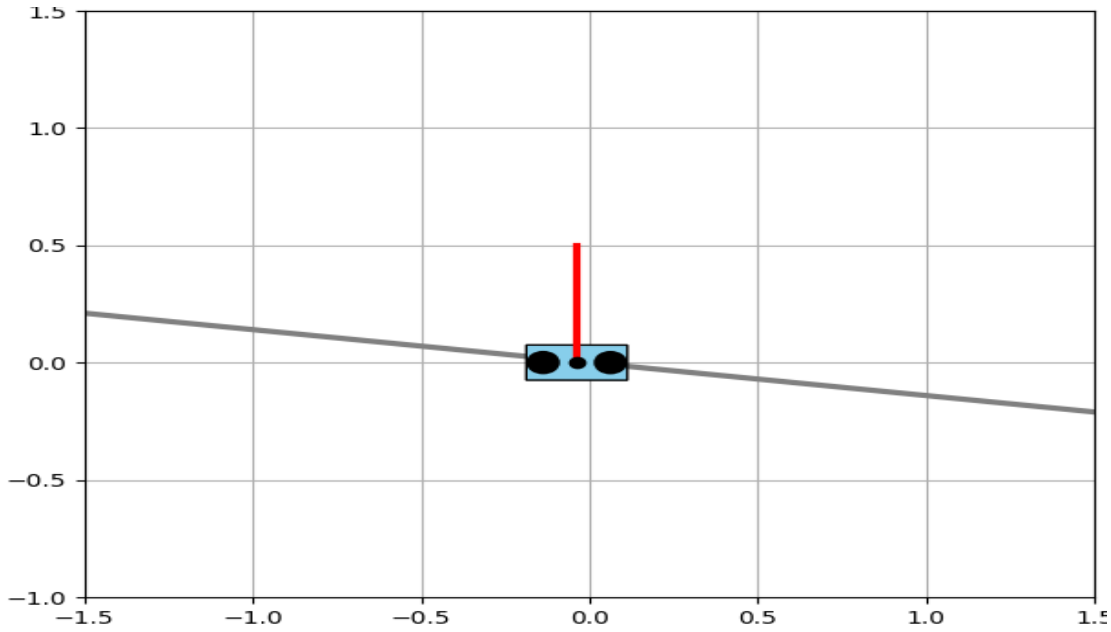

Figura 13. Simulação tarefa 2.2

Código da simulação:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from scipy.integrate import solve_ivp
import math

# Parâmetros
M, m, l, g = 1.0, 0.2, 0.5, 9.81
alpha_min = -8 * np.pi / 180
alpha_max = 8 * np.pi / 180

def alpha_real(s):
    return 0.07 * np.sin(0.5 * s)

# Controlador
Kp_theta, Kd_theta = 100, 20
Kp_s, Kd_s = 1, 2

# Estimadores adaptativos
hat_s_alpha_int, hat_s_alpha_ext, hat_c_alpha = 0.0, 0.0, 1.0
```

```
gamma_s_int, gamma_s_ext, gamma_c = 1.0, 1.0, 1.0

# Dinâmica + adaptação
def dynamics_adapt(t, X):
    hat_s_alpha_int, hat_s_alpha_ext, hat_c_alpha
    s, s_dot, theta, theta_dot = X
    a_real = alpha_real(s)
    theta_error, s_error = theta, s

    u = -Kp_theta*theta_error - Kd_theta*theta_dot - Kp_s*s_error - Kd_s*s_dot

    M11 = M + m
    M12 = -m*l*np.cos(theta)
    M22 = m*l**2
    C1 = m*l*theta_dot**2*np.sin(theta)
    G1 = (M+m)*g*hat_s_alpha_ext
    G2 = -m*g*l*hat_c_alpha*np.sin(theta)

    Mat = np.array([[M11, M12],[M12, M22]])
    B = np.array([u - C1 - G1, -G2])
    s_ddot, theta_ddot = np.linalg.solve(Mat, B)
    hat_s_alpha_int += gamma_s_int * theta_error * 0.01
    hat_s_alpha_ext += gamma_s_ext * s_error * 0.01
    hat_c_alpha += gamma_c * theta_error * np.sin(theta) * 0.01
    hat_s_alpha_int = np.clip(hat_s_alpha_int, np.sin(alpha_min), np.sin(alpha_max))
    hat_s_alpha_ext = np.clip(hat_s_alpha_ext, np.sin(alpha_min), np.sin(alpha_max))
    hat_c_alpha = np.clip(hat_c_alpha, np.cos(alpha_max), np.cos(alpha_min))

    return [s_dot, s_ddot, theta_dot, theta_ddot]

# Simulação
X0 = [0.0, 0.0, 0.01, 0.0]
t_span = (0, 10)
t_eval = np.linspace(*t_span, 500)
sol = solve_ivp(dynamics_adapt, t_span, X0, t_eval=t_eval)
s, theta = sol.y[0], sol.y[2]

# Animação
fig, ax = plt.subplots(figsize=(8, 6))
ax.set_aspect('equal')
```

```
ax.grid(True)

# trilho
track_len = 2.5
line_x = np.linspace(-track_len, track_len, 50)
alpha_est = np.arcsin(np.mean(np.clip(hat_s_alpha_ext, np.sin(alpha_min), np.sin(alpha_max))))
track_y = np.tan(alpha_est) * line_x
ax.plot(line_x, track_y, 'gray', linewidth=3)

# objetos gráficos
carrinho_box = plt.Rectangle((0,0), 0.3, 0.15, fc='skyblue', ec='black')
rod_esq = plt.Circle((0,0), 0.05, fc='black')
rod_dir = plt.Circle((0,0), 0.05, fc='black')
pend_line, = ax.plot([], [], 'r-', lw=3)
pivot_dot, = ax.plot([], [], 'ko')
ax.add_patch(carrinho_box)
ax.add_patch(rod_esq)
ax.add_patch(rod_dir)
ax.set_xlim(-1.5, 1.5)
ax.set_ylim(-1.0, 1.5)
plt.title('Carrinho sobre trilho inclinado com pêndulo ereto para cima')

def init():
    pend_line.set_data([], [])
    pivot_dot.set_data([], [])
    return carrinho_box, rod_esq, rod_dir, pend_line, pivot_dot

def update(i):
    x = s[i]
    θ = theta[i]
    a_real = alpha_real(x)
    ca, sa = math.cos(a_real), math.sin(a_real)

    # posição carrinho sobre trilho inclinado
    Xc = x * ca
    Yc = x * sa

    # caixa do carrinho
    carrinho_box.set_xy((Xc - 0.15, Yc - 0.075))
    carrinho_box.angle = np.degrees(a_real)
```

```
    # rodas
    rod_esq.center = (Xc - 0.1*ca, Yc - 0.1*sa)
    rod_dir.center = (Xc + 0.1*ca, Yc + 0.1*sa)

    # pêndulo ereto para cima: sempre vertical invertido (perpendicular ao chão)
    xp = Xc
    yp = Yc + l  # para cima no mundo, independentemente da inclinação
    pend_line.set_data([Xc, xp], [Yc, yp])
    pivot_dot.set_data(Xc, Yc)
    return carrinho_box, rod_esq, rod_dir, pend_line, pivot_dot
ani = FuncAnimation(fig, update, frames=len(s), init_func=init, blit=True, interval=20)
plt.show()
```

A. 8 - Código MATLAB Utilizado (Exercício 2.1)

```
% Simulação Controle - Controle Fixo x Adaptativo
% Parâmetros
m = 0.3; % kg
M = 1.5; % kg
l = 0.3; % m
g = 9.78; % m/s^2
kv = 1;  % kg/s
dt = 0.01;
T = 10;
N = T/dt;

% Ganhos para controller em cascata (a ajustar)
kp_pos = 5;
kd_pos = 1;
beta_p = 5;
beta_d = 1;

% Limites
theta_max = 8*pi/180; % saturação de referência de ângulo em radianos

% Variáveis iniciais
s = 0; s_dot = 0;
theta = 0; theta_dot = 0;
s_d = 5; % posição desejada

% alpha constante (exemplo)
alpha = 5*pi/180; % constante, pode mudar para função suave de s

% Para armazenar resultados
s_hist = zeros(1,N);
theta_hist = zeros(1,N);
theta_d_hist = zeros(1,N);
u_hist = zeros(1,N);

for k=1:N
   % Controle posição (malha externa)
   e_s = s_d - s;
   e_s_dot = - s_dot;

   % Lei de controle para referência de ângulo (theta_d), saturada
   theta_d = kp_pos*e_s + kd_pos*e_s_dot;
   theta_d = max(min(theta_d, theta_max), -theta_max);
      % Controle ângulo (malha interna)
   e_theta = theta_d - theta;
   e_theta_dot = -theta_dot;
      u = beta_p*e_theta + beta_d*e_theta_dot; % sinal de controle
      % Dinâmica simplificada do sistema (modelo linear aproximado)
   s_ddot = (u - kv*s_dot)/(M + m);
   theta_ddot = (g/l)*theta + s_ddot * cos(alpha); % linearizada

   % Integração numérica (Euler)
   s_dot = s_dot + s_ddot*dt;
   s = s + s_dot*dt;
   theta_dot = theta_dot + theta_ddot*dt;
   theta = theta + theta_dot*dt;

   % Armazenar dados
   s_hist(k) = s;
```

```
    theta_hist(k) = theta;
    theta_d_hist(k) = theta_d;
    u_hist(k) = u;

    % Atualizar alpha se quiser função suave (exemplo)
    % alpha = 8*pi/180 * sin(pi*s/T); % exemplo variável
end

time = (0:N-1)*dt;

% Gráficos
figure;
subplot(3,1,1);
plot(time, s_hist);
xlabel('Tempo (s)'); ylabel('Posição s (m)');
title('Posição do Carrinho');

subplot(3,1,2);
plot(time, theta_hist*180/pi);
hold on;
plot(time, theta_d_hist*180/pi, 'r--');
xlabel('Tempo (s)'); ylabel('Ângulo (°)');
legend('\theta', '\theta_d');
title('Ângulo do Pêndulo e Referência');

subplot(3,1,3);
plot(time, u_hist);
xlabel('Tempo (s)'); ylabel('Controle u');
title('Sinal de Controle');
```

A.9 – Exercício 2.2 – Implementação Matlab

Código:

```
% Simulação Controle - Controle Fixo x Adaptativo
clear; close all; clc;

% Parâmetros da simulação
dt = 0.01;          % passo de tempo (s)
T = 10;             % tempo total (s)
N = T/dt;           % número de iterações

% Ganhos PID do controlador angular (C_theta)
Kp_theta = 5;
Ki_theta = 0.1;
Kd_theta = 0.01;

% Ganho controlador posição (C_s) - fixo (não usado direto na adaptação)
alpha_real = 8*pi/180 * sin(linspace(0,pi,N)); % alpha real variável no tempo (exemplo)
% alpha_real varia entre -8° e +8°, simula a inclinação variável do trilho

% Velocidade linear constante
v = 1; % m/s

% Posição desejada
s_d = 5; % metros

% --- Inicialização variáveis fixo ---
s_fix = 0; theta_fix = 0;
```

```
integral_e_theta_fix = 0; prev_e_theta_fix = 0;

% --- Inicialização variáveis adaptativo ---
s_adapt = 0; theta_adapt = 0;
integral_e_theta_adapt = 0; prev_e_theta_adapt = 0;
hat_alpha = 0;         % estimativa inicial de alpha (rad)
gamma_alpha = 10;      % ganho adaptativo

% Armazenar dados para fixo
s_hist_fix = zeros(1,N);
theta_hist_fix = zeros(1,N);
e_s_hist_fix = zeros(1,N);

% Armazenar dados para adaptativo
s_hist_adapt = zeros(1,N);
theta_hist_adapt = zeros(1,N);
e_s_hist_adapt = zeros(1,N);
hat_alpha_hist = zeros(1,N);

for k = 1:N
    % --- Controle FIXO ---
    e_s_fix = s_d - s_fix;
    alpha_k = alpha_real(k);   % alpha real variável no tempo
    theta_d_fix = alpha_k * e_s_fix;

    e_theta_fix = theta_d_fix - theta_fix;
    integral_e_theta_fix = integral_e_theta_fix + e_theta_fix*dt;
    deriv_e_theta_fix = (e_theta_fix - prev_e_theta_fix)/dt;

    omega_fix = Kp_theta*e_theta_fix + Ki_theta*integral_e_theta_fix + Kd_theta*deriv_e_theta_fix;

    theta_fix = theta_fix + omega_fix*dt;
    s_fix = s_fix + v*cos(theta_fix)*dt;

    prev_e_theta_fix = e_theta_fix;

    s_hist_fix(k) = s_fix;
    theta_hist_fix(k) = theta_fix;
    e_s_hist_fix(k) = e_s_fix;

    % --- Controle ADAPTATIVO ---
    e_s_adapt = s_d - s_adapt;
    theta_d_adapt = hat_alpha * e_s_adapt;

    e_theta_adapt = theta_d_adapt - theta_adapt;
    integral_e_theta_adapt = integral_e_theta_adapt + e_theta_adapt*dt;
    deriv_e_theta_adapt = (e_theta_adapt - prev_e_theta_adapt)/dt;

    omega_adapt = Kp_theta*e_theta_adapt + Ki_theta*integral_e_theta_adapt +
Kd_theta*deriv_e_theta_adapt;

    theta_adapt = theta_adapt + omega_adapt*dt;
    s_adapt = s_adapt + v*cos(theta_adapt)*dt;

    prev_e_theta_adapt = e_theta_adapt;

    % Atualiza estimativa adaptativa de alpha
    hat_alpha_dot = gamma_alpha * e_s_adapt * e_theta_adapt;
    hat_alpha = hat_alpha + hat_alpha_dot * dt;
    % Saturação
```

```
    hat_alpha = max(min(hat_alpha, 8*pi/180), -8*pi/180);

    s_hist_adapt(k) = s_adapt;
    theta_hist_adapt(k) = theta_adapt;
    e_s_hist_adapt(k) = e_s_adapt;
    hat_alpha_hist(k) = hat_alpha;
end

% Tempo para plot
time = (0:N-1)*dt;

% --- Gráficos comparativos ---
figure('Name','Comparação Controle Fixo x Adaptativo','Position',[100 100 900 900]);
% Posição
subplot(4,1,1);
plot(time, s_hist_fix,'b','LineWidth',1.5); hold on;
plot(time, s_hist_adapt,'r--','LineWidth',1.5);
yline(s_d,'k--','LineWidth',1);
xlabel('Tempo (s)');
ylabel('Posição s (m)');
title('Posição Linear: Controle Fixo (azul) x Adaptativo (vermelho)');
legend('Fixo','Adaptativo','Posição Desejada');
grid on;
% Erro de posição
subplot(4,1,2);
plot(time, e_s_hist_fix,'b','LineWidth',1.5); hold on;
plot(time, e_s_hist_adapt,'r--','LineWidth',1.5);
xlabel('Tempo (s)');
ylabel('Erro de Posição (m)');
title('Erro de Posição');
legend('Fixo','Adaptativo');
grid on;
% Ângulo θ
subplot(4,1,3);
plot(time, theta_hist_fix,'b','LineWidth',1.5); hold on;
plot(time, theta_hist_adapt,'r--','LineWidth',1.5);
xlabel('Tempo (s)');
ylabel('Ângulo \theta (rad)');
title('Ângulo do Robô');
legend('Fixo','Adaptativo');
grid on;
% Estimativa adaptativa de alpha
subplot(4,1,4);
plot(time, hat_alpha_hist*180/pi,'r','LineWidth',1.5); hold on;
plot(time, alpha_real*180/pi,'b--','LineWidth',1.5);
xlabel('Tempo (s)');
ylabel('α (graus)');
title('Estimativa Adaptativa \hat{\alpha} vs α real');
legend('\hat{\alpha}','\alpha real');
grid on;
```