Universidade Federal do Rio Grande do Norte

EGM0001 - SISTEMAS DINÂMICOS E SERVOMECANISMOS

Atividade Computacional 03

Controle de Sistema de 1ª Ordem

Discente:
Camila Barbosa Gomes de Araújo

Docente:
Wallace Moreira Bessa

19 de dezembro de 2020

1 Modelo proposto

O sistema a ser controlado trata-se do subsistema Motor-Roda do Rover Perseverance. O modelo dinâmico proposta para esse subsistema está descrito pela Equação 1. Para esse problema, no entanto, está sendo suposto que os coeficientes de atrito, μ_v e μ_d sejam constantes e bastante próximos, de modo que seja possível assumir que $\mu_v = \mu_d = \mu = cte$. Assim, tem-se:

$$J\dot{\omega} = -\mu(\omega + sgn(\omega)) + Ki \tag{1}$$

Onde:

- J: Inércia
- ω: Velocidade angular
- μ: Coeficiente de Atrito
- K: Constante do Motor
- i: Corrente

Para fins de implementação, serão considerados:

- $J = 2 \times 10^{-4}$
- $K = 4 \times 10^{-2}$

Dadas essas considerações, agora está sendo considerada a lei de controle descrita na Equação 2.

$$i = \frac{\hat{\mu}(\omega + sgn(\omega) + J(\dot{\omega} - \lambda e)}{K} \tag{2}$$

Sendo:

- $e = \omega - \omega_d \longrightarrow$ Erro
- $\lambda \longrightarrow$ Lambda (Coeficiente de aprendizado)

O parâmetro $\hat{\mu}$ é calculado a cada instante de atuação do controlador pela lei adaptativa descrita na Equação 3.

$$\hat{\mu}_{n+1} = \hat{\mu}_n - \eta e_n(\omega_n + sgm(\omega_n))\Delta t \tag{3}$$

Nas seções seguintes, serão tratadas a implementação do subsistema em Python, os resultados com gráficos do comportamento do sistema e as devidas conclusões sobre o experimento.

2 Implementação do Sistema

Aproveitando-se da implementação da primeira atividade computacional, o novo sistema proposto foi desenvolvido com algumas alterações:

1. Mudança na lei de controle (Equação 2)
2. Mudança do valor calculado de $\dot{\omega}$ (Equação 1)

3. Cálculo de $\hat{\mu}$ (Equação 3)

No trecho de código abaixo, a função principal do código é exposta, com as devidas alterações comentadas acima.

```
def generate_values(tf, W, Wd, _lambda, J, K, mi, u_chapeu, N):
    _w = [W]
    _e = []
    _i = []
    _u_chapeu = [0]
    for t in range(tf*1000-1):
        e = W - Wd
        _e.append(e) ### save the error to plot later
        i = (u_chapeu*(W+np.sign(W)) + J*(0-(_lambda*e))) / K ### calculates the current
        _i.append(i)
        u_chapeu = u_chapeu - (N * e * (W + np.sign(W)) * 0.001)
        _u_chapeu.append(u_chapeu)
        W = rungeKutta(W, 0.001, mi, K, i, J, u_chapeu)
        _w.append(W) ### saves in a list the next value of w
    _e.append(W - Wd)
    _i.append((u_chapeu*(W+np.sign(W)) + J*(0-(_lambda*e))) / K)
    return _w, _e, _i, _u_chapeu
```

O código pode ser encontrado em `https://github.com/camilabga/rover-control`. A função que implementa essa atividade é *plot* em *main03.py* e *functions03.py*.

3 Resultados e Conclusões

A análise consistiu em variar η e μ e entender os resultados. Foram utilizados dois valores de μ e dois valores de λ com o η variando sempre entre 7 valores.

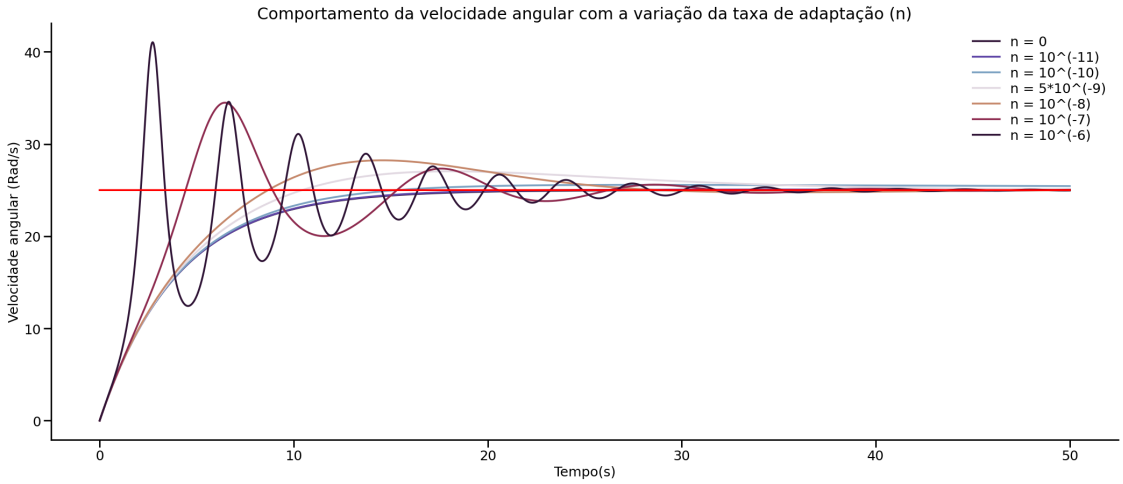

Figura 1: Plot da velocidade angular com a variação de η com $\mu = 0$ e $\lambda = 0.25$

O primeiro teste consistiu em analisar o comportamento da velocidade angular, da corrente e do valor

estimado do coeficiente de atrito para 7 valores diferentes da taxa de adaptação, quando $\mu = 0$ e $\lambda = 0.25$. Os gráficos estão nas Figuras 1, 2 e 3.

A princípio, percebemos que ao aumentar o η, também aumentamos a oscilação na velocidade angular. Isso pode ser visto na Figura 1. Com a taxa de adaptação nula, ou quase nula, percebemos uma curva suave e sem (ou com muito pouco) *overshoot*.

Na Figura 2, o esforço de controle repete o visto na velocidade angular. Quanto maior o valor de η, mais oscilatório o comportamento.

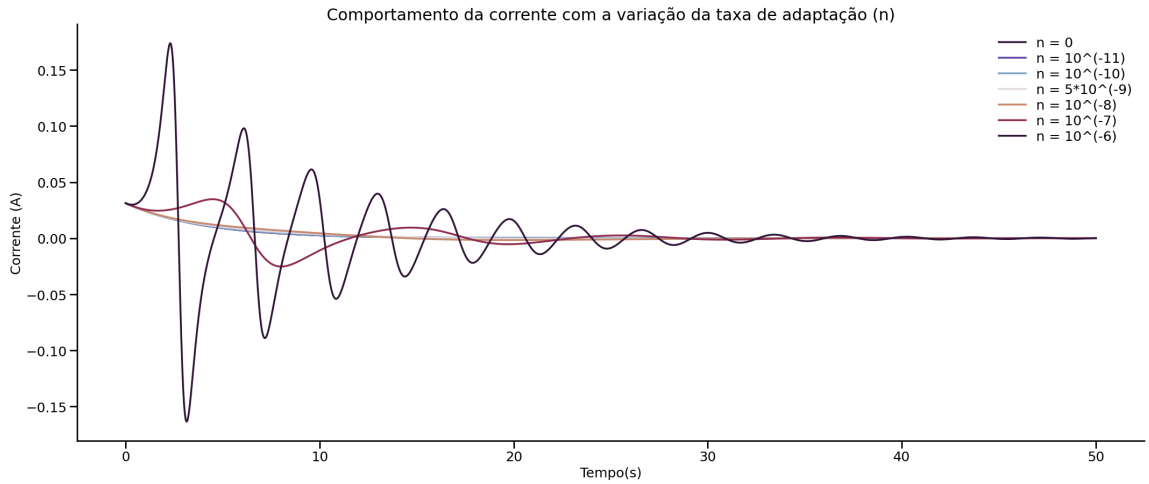

Figura 2: Plot da corrente com a variação de η com $\mu = 0$ e $\lambda = 0.25$

Já comportamento da estimativa do coeficiente de atrito, por o mesmo ser igual a zero, nesse caso, o valor estimado oscila sobre o zero até estabilizar quanto maior a taxa de adaptação, o que faz sentido. Enquanto para os valores menores, a convergência com o valor real do modelo é mais rápida, o que pode ser visto na Figura 3

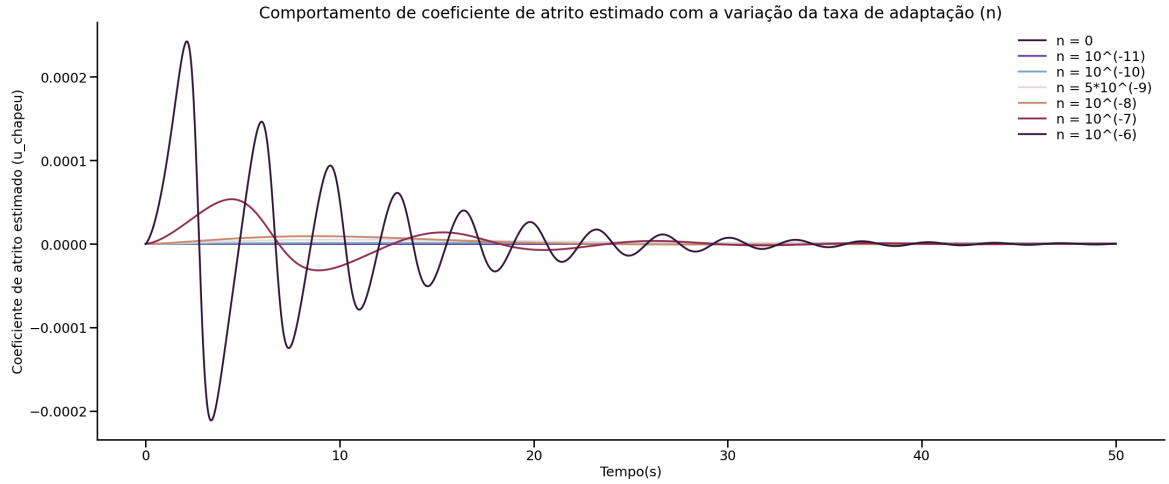

Figura 3: Plot da estimativa do coeficiente de atrito com a variação de η com $\mu = 0$ e $\lambda = 0.25$

Ao variar o valor do coeficiente de atrito μ do modelo para 0.0005 e mantendo o tempo de execução em 50 segundos, temos os resultados das Figuras 4 e 5.

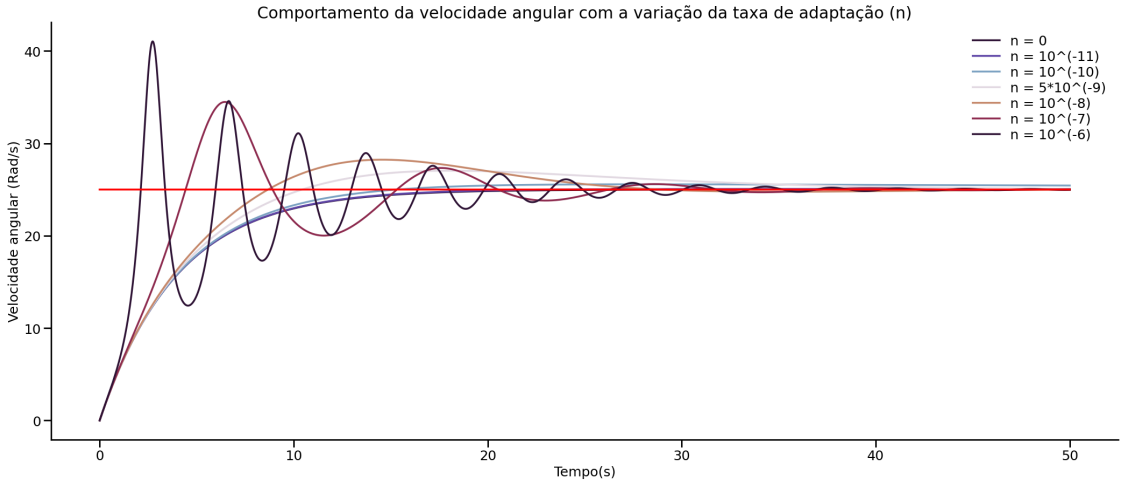

Figura 4: Plot da velocidade angular com a variação de η com $\mu = 0.0005$ e $\lambda = 0.25$

Na Figura 4, notamos que apenas a simulação com a maior taxa de adaptação foi capaz de convergir em menos de 50 segundos. Isso acontece porque o coeficiente de atrito do modelo agora é diferente de zero e a estimativa do mesmo, que é o valor considerado na lei de controle, depende diretamente de η para variar.

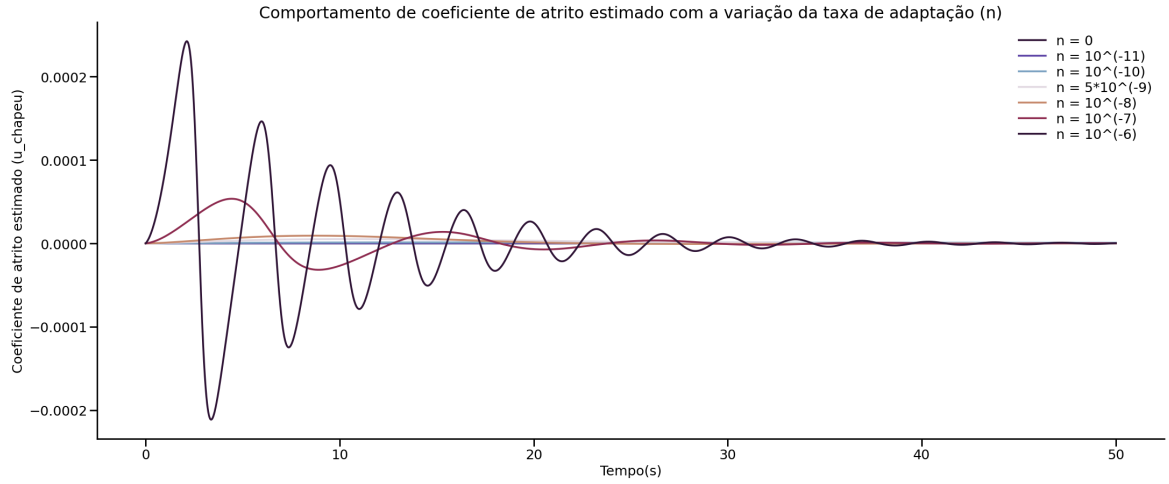

Figura 5: Plot da estimativa do coeficiente de atrito com a variação de η com $\mu = 0.0005$ e $\lambda = 0.25$

Na Figura 5 notamos exatamente isso, o valor com maior taxa de adaptação oscilando e os seguintes se aproximando de $\mu = 0.0005$.