Relatório do Exame de CMC-12

Uso de Técnicas de Aprendizado de Máquina para Controle de Sistemas Dinâmicos

Bruno Franco Vieira
Fellipe Mascarenhas
Raul Teodoro Diniz Silveira

COMP 26

Professor:
Marcos Ricardo Omena de Albuquerque Maximo

Instituto Tecnológico de Aeronáutica – ITA

Sumário

1. Fundamentação Teórica

- Neste projeto, buscou-se solicionar o problema do *CartPole*, encontrado no site Gymnasium, utilizando um controlador proporcional-integrativo-derivativo (PID) de ganhos variáveis que evoluem com a técnica de *Particle Swarm Optimization* (PSO).
- Como neste problema a solução envolve a posição x do carro e o ângulo θ do mastro, foram criados dois controladores PID.

1.1. Controlador PID

- Como visto em aula, temos que a lei de controle do controlador PID é:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \dot{e}(t) \tag{1}$$

- Os erros no momento t são dados por:

$$e_\theta(t) = \theta_r - \theta(t) \tag{2}$$

$$e_x(t) = x_r - x(t) \tag{3}$$

- Considerando que a taxa de atualização Δt é pequena o suficiente, fazemos que:

$$\int_0^t e(\tau)\, d\tau \approx \Sigma_0^t \tau \Delta t \tag{4}$$

$$\dot{e}(t) \approx \frac{e(t) - e(t-1)}{\Delta t} \tag{5}$$

1.2. *Particle Swarm Optimization*

- A implementação seguiu a teoria de PSO lecionada na eletiva CT-213.
- Como o objetivo é determinar os ganhos que descrevem um bom comportamento do carro, são criadas duas populações de partículas, que determinam os ganhos para controlar a posição do carro e o ângulo do mastro.
- Ambas as populações apresentam 40 partículas e evoluem em 10000 episódios, além de utilizarem os mesmos hiperparâmetros.
- Inicialização da velocidade da partícula:

$$v_i \sim U[l - u, u - l] \tag{6}$$

- Atualização da velocidade da partícula:

$$v_i = \omega v_i + \varphi_p r_p (b_i - x_i) + \varphi_g r_g (b_g - x_i) \tag{7}$$

- Os valores r_p e r_g são randômicamente escolhidos em uma distribuição normal de 0 a 1 em toda atualiação.
- Inicialização da posição da partícula:

$$x_i \sim U[l,u] \tag{8}$$

- Atualização da posição da partícula:

$$x_i = x_i + v_i \tag{9}$$

- A heurística ultilizada é limitar posição e velocidade após o cálculo na iteração, logo:

$$x_i = min(max(x_i,l),u)$$
$$v_i = min(max(v_i, l-u), u-l)$$

- Equação da medida de qualidade é cumulativa, sendo que a recompensa é obtida pelo próprio ambiente da API do Gymnasium (*CartPole*), seguindo a seguinte fórmula:

$$f(x) = \Sigma_{\mathrm{k=1}}^{N} reward_k \tag{10}$$

2. Implementação Desenvolvida

2.1. Constantes

- Em um arquivo são definidas as constantes, que são:

1. $\theta_r = 0$
2. $x_r = 0$
3. $\Delta t = 1$
4. Número de episódios: 10000

A escolha do $\Delta t = 1$ foi precisa para ser semelhante ao passo do ambiente de simulação da API do Gymnasium, caso fosse escolhido um tempo menor, a solução não convergiria bem.

2.2. Parâmetros

- É criada uma classe *Params* apenas para facilitar a conversão da posição da partícula nos ganhos.

2.3. Controlador PID

- É criada um classe responsável pelo controle das ações do carro e do poste.
- Esta classes calcula o termo integrativo segundo a equação 4 , o termo derivativo segundo a equação 5 e a saída do controlador segundo a equação 1.
- A classe também apresenta um método estático que calcula os erros utilizando as equações 2 e 3, em que a referência tanto da posição quando do ângulo é 0, depois, usando os próprios controladores de ângulo e posição, decide-se a ação. Como os ganhos são positivos, se $u_\theta(t) + u_x(t) < 0$, o comportamento é de o ângulo do mastro estar aumentando, logo a ação deve ser de mover o carro para a direita. O contrário indica que o carro deve ser movido para a esquerda.

2.4. *Particle Swarm Optimization*

- É criada uma classe para as partículas, que inicia a posição e a velocidade segundo as equações 6 e 8, respectivamente.
- Os limites dos ganhos para cada controlador, que foram escolhidos empiricamente, são:

Para o controlador horizontal

1. $l_p = 100$
2. $u_p = 120$
3. $l_i = 0.1$
4. $u_i = 0.3$
5. $l_d = 0.8$
6. $u_d = 2.0$

Para o controlador do ângulo

1. $l_p = 100$
2. $u_p = 110$
3. $l_i = 0.3$
4. $u_i = 0.5$
5. $l_d = 4.0$
6. $u_d = 6.0$

- Uma classe para toda a população é criada, armazenando a melhor posição da geração, a melhor posição global, avançando as gerações segundo as equações 7 e 9 e avaliando a geração.
- Os valores dos hiper parâmetros foram definidos com base em funções que ajustam seu valor linearmente em função da iteração para que tenha-se tanto explotação quanto exploração, os valores do peso de inércia, parâmetro cognitivo e social são :

1. $\omega_{ini} = 0.3$ e $\omega_{fim} = 1$
2. $\varphi_p^{init} = 1$ e $\varphi_p^{fim} = 0.3$
3. $\varphi_g^{init} = 1$ e $\varphi_g^{fim} = 0.4$

A escolha desses parâmetros foi totalmente empírica, a ideia seria começar com alta exploração (altos parâmetros cognitivos e sociais e baixo peso de inércia) e em seguida aumentar a explotação (baixos parâmetros cognitivos e sociais e alto peso de inércia).

2.5. Implementação completa

- Inicialmente, criam-se e iniciam-se os controladores do ângulo e da posição, além da população para o uso do PSO para determinar os ganhos.
- São usados novos valores de ganhos e então é renderizado o ambiente, tomando até 500 ações com os novos ganhos, calculando e acumulando as recompensas, segundo a equação 10. O resultado então é armazenado e o desempenho é avaliado pelo PSO.
- Após cada simulação é feita a atualização do melhor valor e o avanço de geração.
- Estes processos, com exceção do inicial, são repetidos em todos os episódios.

3. Resultados e Discussões

Com base nos gráficos apresentados, é possível realizar uma análise detalhada dos resultados obtidos durante a otimização dos controladores PID utilizando o algoritmo de Particle Swarm Optimization (PSO). O objetivo principal é avaliar a eficiência do PSO na sintonia dos parâmetros K_p, K_i e K_d dos controladores PID e sua convergência para soluções ótimas.

3.1. Convergência do Tempo

Os gráficos de convergência do tempo, tanto do melhor tempo quanto do tempo de simulação, demonstram que os valores se estabilizaram rapidamente. No gráfico de "Best Time Convergence" (Figura 1), observa-se que o melhor tempo atingiu rapidamente um valor constante de 500, indicando que o controlador encontrou uma solução ótima em poucas iterações. Esse comportamento é esperado quando o PSO encontra rapidamente uma região promissora no espaço de busca.

O PSO, inspirado no comportamento coletivo de pássaros e peixes, é eficaz na exploração global e na exploração local, o que resulta em uma rápida convergência. O algoritmo ajusta dinamicamente a posição de cada partícula (solução candidata) com base em sua própria experiência e na experiência de suas vizinhas, permitindo encontrar rapidamente a região ótima no espaço de busca.

Segundo Ogata, métodos de otimização como o PSO são essenciais para a sintonia de controladores, especialmente em sistemas onde métodos tradicionais de tentativa e erro não são eficientes [1].

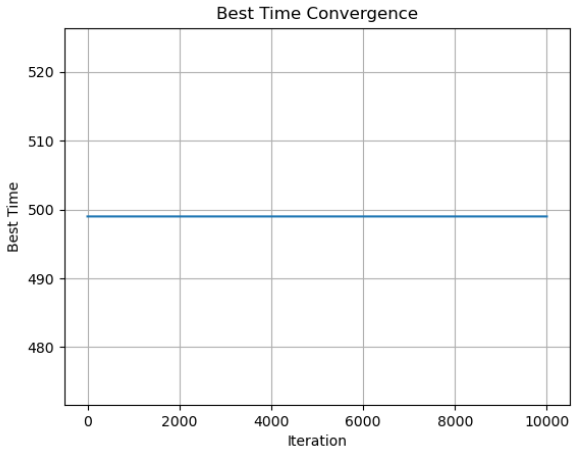

Figura 1: Best Time Convergence

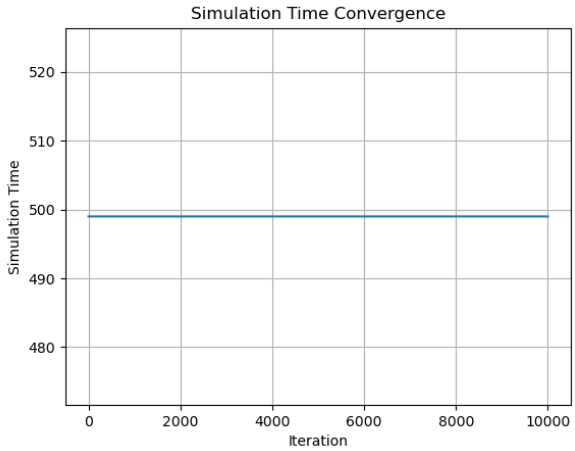

Figura 2: Simulation Time Convergence

O gráfico de "Simulation Time Convergence" (Figura 2) apresenta um comportamento semelhante, com o tempo de simulação convergindo para o valor de 500 em poucas iterações. Esse comportamento confirma a eficácia do PSO em encontrar uma solução ótima de forma eficiente, minimizando o tempo necessário para alcançar uma resposta estável.

3.2. Convergência dos Parâmetros para Controle de Ângulo (Theta) e Controle Horizontal

Os gráficos de convergência dos parâmetros do controlador PID para o controle de ângulo (Figura 3) e controle horizontal (Figura 4) mostram que os valores dos ganhos K_p, K_i e K_d se estabilizaram ao longo das iterações.

Para o controle de ângulo (Figura 3), observa-se que o ganho proporcional K_p converge para um valor próximo de 100, enquanto os ganhos integrativo K_i e derivativo K_d convergem para valores

próximos de 0.1 e 5, respectivamente. Esse comportamento indica que o controle de ângulo requer um ganho proporcional elevado para corrigir rapidamente os erros angulares, enquanto os ganhos integrativo e derivativo ajudam a estabilizar o sistema.

O ganho proporcional K_p é responsável pela correção imediata do erro, o ganho integrativo K_i corrige erros acumulados ao longo do tempo, e o ganho derivativo K_d prevê futuras tendências de erro com base na taxa de variação do erro. A sintonia adequada desses ganhos é crucial para um desempenho ótimo do sistema de controle [2].

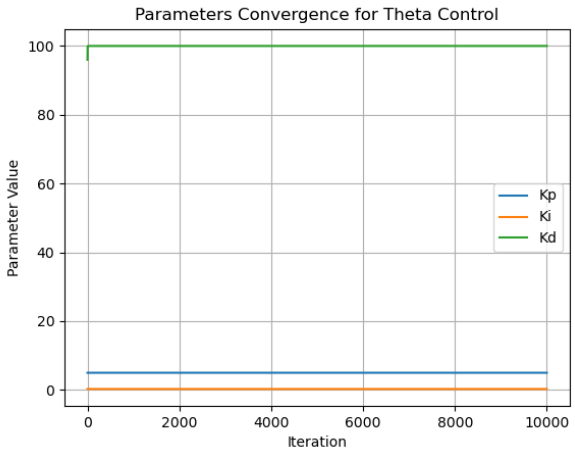

Figura 3: Convergência dos parâmetros do controlador PID para o controle de ângulo

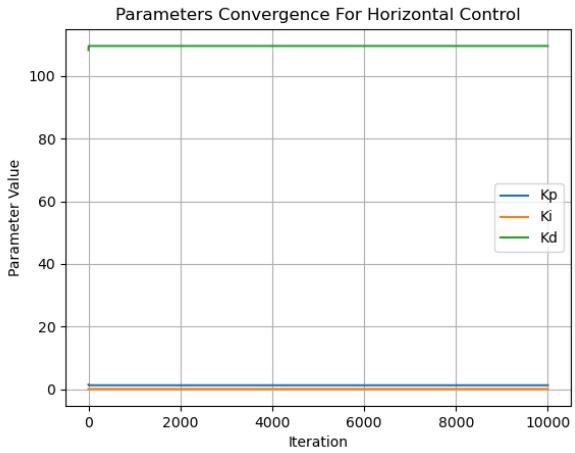

Figura 4: Convergência dos parâmetros do controlador PID para o controle horizontal

No caso do controle horizontal (Figura 4), o ganho proporcional K_p converge para um valor próximo de 110, enquanto os ganhos K_i e K_d se estabilizam em valores próximos de 0.5 e 3, respectivamente. Esses valores sugerem que o controle horizontal necessita de um ganho proporcional moderado para corrigir a posição do carro, com contribuições menores dos termos integrativo e derivativo.

A análise dos gráficos de convergência dos parâmetros PID confirma que o PSO foi eficaz em ajustar os ganhos para otimizar a resposta do sistema. A estabilidade dos valores ao longo das iterações é um indicativo de que o algoritmo encontrou uma configuração robusta para os controladores PID.

3.3. Gráficos do movimento do carrinho e do poste

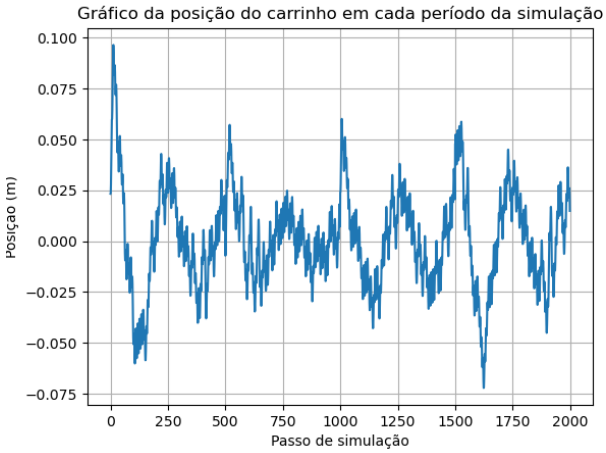

Figura 5: Posição do carrinho com PID otimizado.

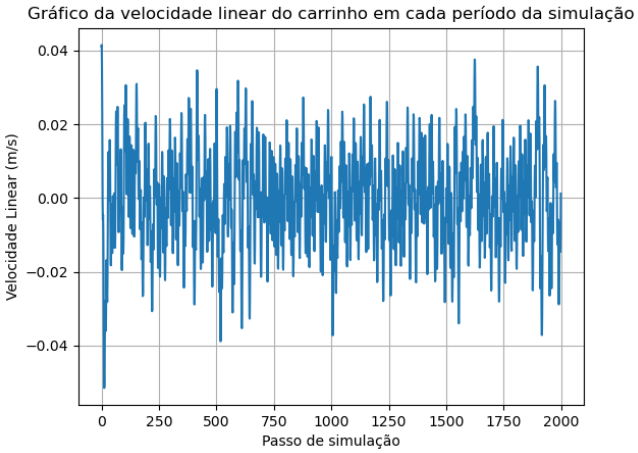

Figura 6: Velocidade do carrinho com PID otimizado.

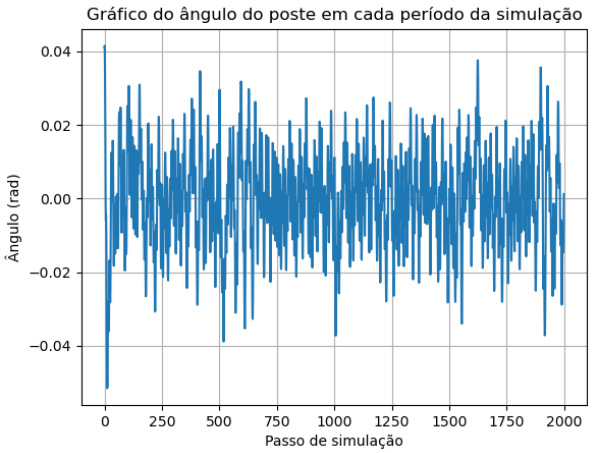

Figura 7: Ângulo do poste com PID otimizado.

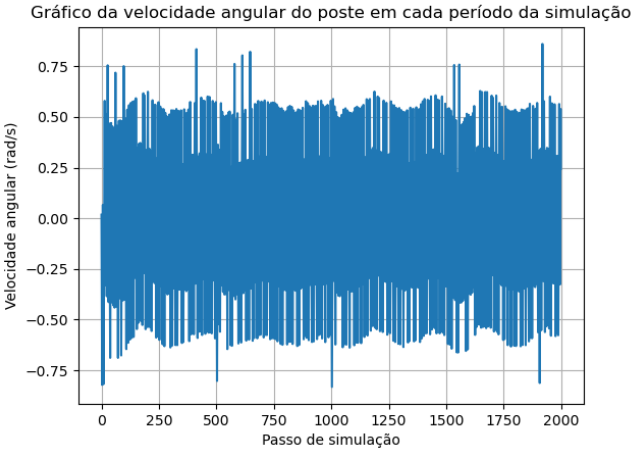

Figura 8: Velocidade angular do poste com PID otimizado.

3.3.1. Posição do Carrinho

O gráfico da Figura 5 mostra a variação da posição do carrinho ao longo da simulação. A posição do carrinho oscila em torno de 0, variando entre aproximadamente -0.075 e 0.075 metros. Essas oscilações indicam que o sistema está tentando manter a posição estável, mas há uma variação constante, sugerindo uma instabilidade que lembra um Movimento Harmônico Simples (MHS) com um pouco de ruído.

3.3.2. Velocidade do Carrinho

O gráfico da Figura 6 apresenta a variação da velocidade linear do carrinho. A velocidade oscila, variando entre aproximadamente -0.04 e 0.04 m/s. Assim como na posição do carrinho, essas flutuações sugerem um esforço constante do sistema para manter a estabilidade, mas com

variações significativas, indicando um comportamento oscilatório semelhante a um MHS com um ruído considerável.

3.3.3. Ângulo do Poste

O gráfico da Figura 7 mostra a variação do ângulo do poste em radianos. O ângulo oscila significativamente acima e abaixo da linha zero, variando entre aproximadamente -0.04 e 0.04 radianos. Essas oscilações indicam que o sistema está tentando equilibrar o poste, mas há variação constante no ângulo, caracterizando um comportamento oscilatório típico de um MHS com bastante ruído.

3.3.4. Velocidade Angular do Poste

O gráfico da Figura 8 apresenta a variação da velocidade angular do poste. As flutuações na velocidade angular, variando entre aproximadamente -0.75 e 0.75 rad/s, indicam um comportamento oscilatório contínuo. Esse comportamento sugere que o sistema está tentando estabilizar a velocidade angular do poste, refletindo ajustes contínuos dos parâmetros do controlador PID.

4. Conclusão

Através dos resultados apresentados, é possível concluir que o uso do algoritmo de Particle Swarm Optimization (PSO) para otimizar os ganhos dos controladores PID foi eficaz em encontrar soluções ótimas para o problema do *CartPole*. Os gráficos de convergência do tempo demonstram que o PSO encontrou rapidamente uma solução ótima, enquanto os gráficos de convergência dos parâmetros indicam que os ganhos dos controladores se estabilizaram em valores apropriados para garantir o desempenho desejado.

A eficácia do PSO como uma ferramenta de otimização se deve à sua capacidade de balancear exploração e exploração, garantindo uma busca global eficiente enquanto refina localmente as melhores soluções encontradas. Por tudo isso, o PSO pode ser uma ferramenta poderosa para a otimização de sistemas de controle, permitindo a obtenção de controladores PID com desempenho superior em sistemas dinâmicos complexos.

Referências

[1] K. Ogata, *Modern Control Engineering*. Prentice Hall, 2010.

[2] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Pearson, 2015.

5. Manual do usuário

É necessário ter instalado o pip ou algum outro gerenciador de pacotes python e o próprio python e rodar os seguintes comandos na raiz do projeto:

```
~$ pip install -r requirements.txt
```

```
~$ python main.py
```