Instituto Tecnológico de Aeronáutica - ITA
CT-213 - Inteligência Artificial aplicada à Robótica Móvel
Aluno: Ulisses Lopes da Silva

Relatório do Laboratório 4 - Otimização com Métodos Baseados em População

---

1 Breve Explicação em Alto Nível da Implementação

1.1 Particle Swarm Optimization

A implementação do algoritmo *Particle Swarm Optimization* (PSO) foi baseada no pseudocódigo discutido em aula, adaptado para uma estrutura orientada a objetos, conforme exigido no escopo deste laboratório. O código foi organizado em duas classes principais: `Particle`, responsável por representar uma partícula individual do enxame, e `ParticleSwarmOptimization`, que gerencia toda a lógica do processo de otimização.

Cada partícula foi definida com base em três atributos principais: sua posição atual (representando o vetor de parâmetros a ser otimizado — neste caso, a velocidade linear, Kp, Ki e Kd), sua velocidade de deslocamento no espaço de busca e o melhor valor de desempenho já alcançado por ela (`best_value`). Por sua vez, a classe PSO mantém o controle do melhor valor global já encontrado (`best_global_value`) e da posição correspondente a esse valor (`best_global_pos`). Além disso, essa classe controla o progresso do algoritmo ao longo das gerações e acompanha qual partícula está sendo avaliada no momento.

Durante a execução, a cada episódio avaliado, o método `notify_evaluation()` é invocado com a recompensa acumulada obtida. Esse método verifica se a nova posição da partícula oferece um desempenho superior ao anterior, atualizando os atributos `best_pos` e `best_value` quando necessário. Caso o novo valor seja também melhor do que o melhor valor global encontrado até então, os valores globais são igualmente atualizados. Após todas as partículas de uma geração serem avaliadas, o método `advance_generation()` é chamado, sendo responsável por atualizar as velocidades e posições das partículas com base na equação clássica do PSO, que incorpora os componentes de inércia, aprendizado individual (componente cognitivo) e aprendizado social.

Foi garantido que os parâmetros de cada partícula permanecessem dentro dos limites estabelecidos pelo problema, utilizando-se a função `np.clip`. Da mesma forma, a velocidade de cada partícula foi limitada para evitar variações excessivas de uma iteração para outra. O controle sobre qual partícula está sendo avaliada foi feito por meio da variável `current_index`, assegurando que cada partícula seja avaliada apenas uma vez por geração.

De modo geral, a implementação mostrou-se eficaz, apresentando uma boa capacidade de convergência para o ótimo da função objetivo e otimizando de maneira satisfatória os parâmetros do controlador do robô. Para garantir um tempo adequado de convergência, optou-se por manter o algoritmo em execução por aproximadamente 4 mil iterações, o que permitiu alcançar trajetórias bastante satisfatórias durante a simulação.

2 Figuras Comprovando Funcionamento do Código

2.1 Teste do *Particle Swarm Optimization*

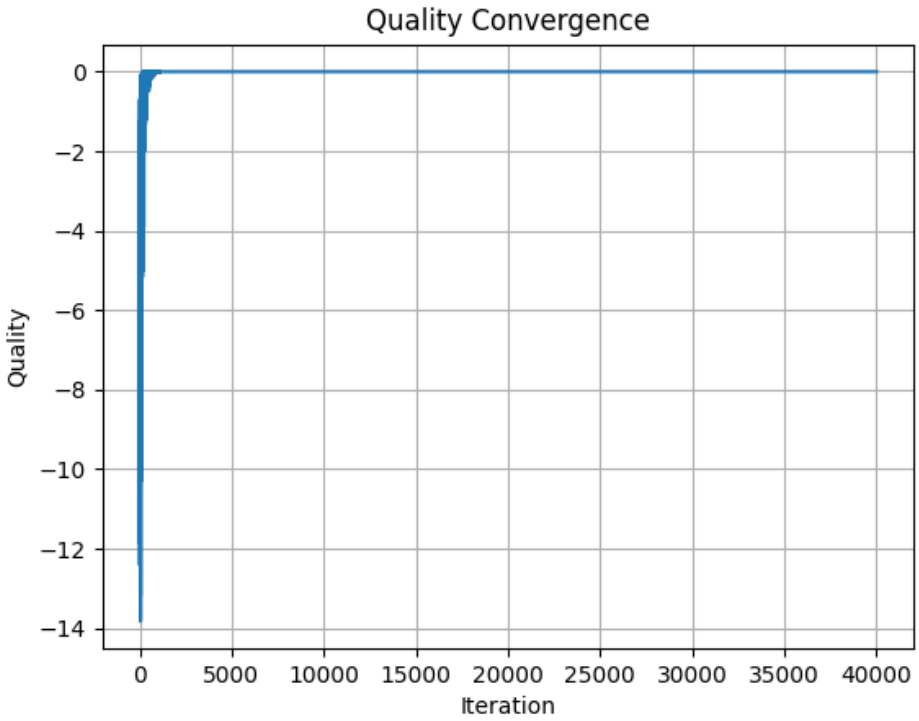

Fig. 1: Teste da qualidade da convergência

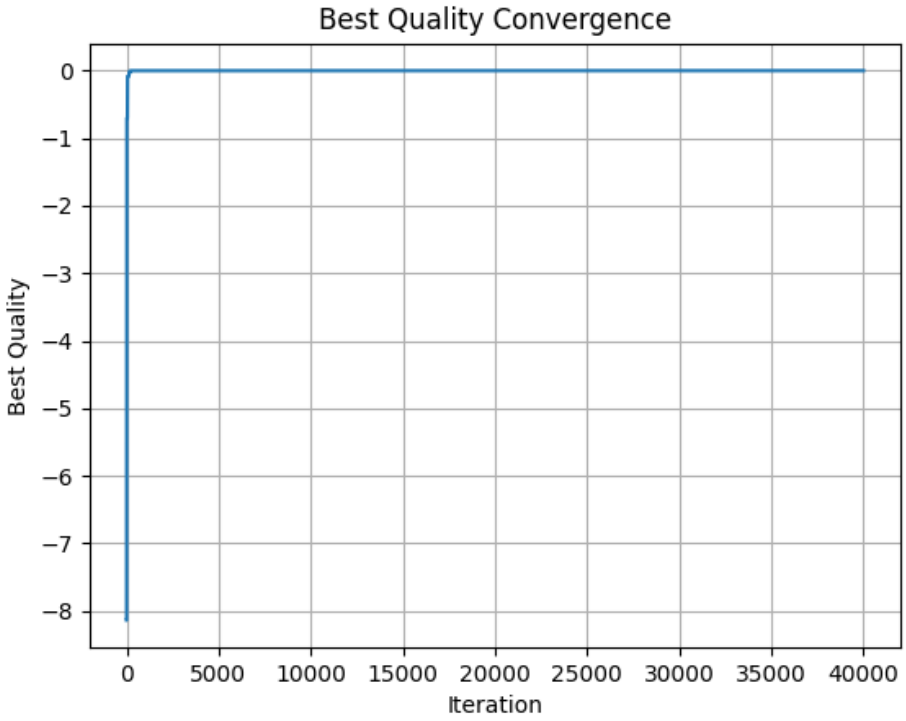

Fig. 2: Teste da melhor convergência

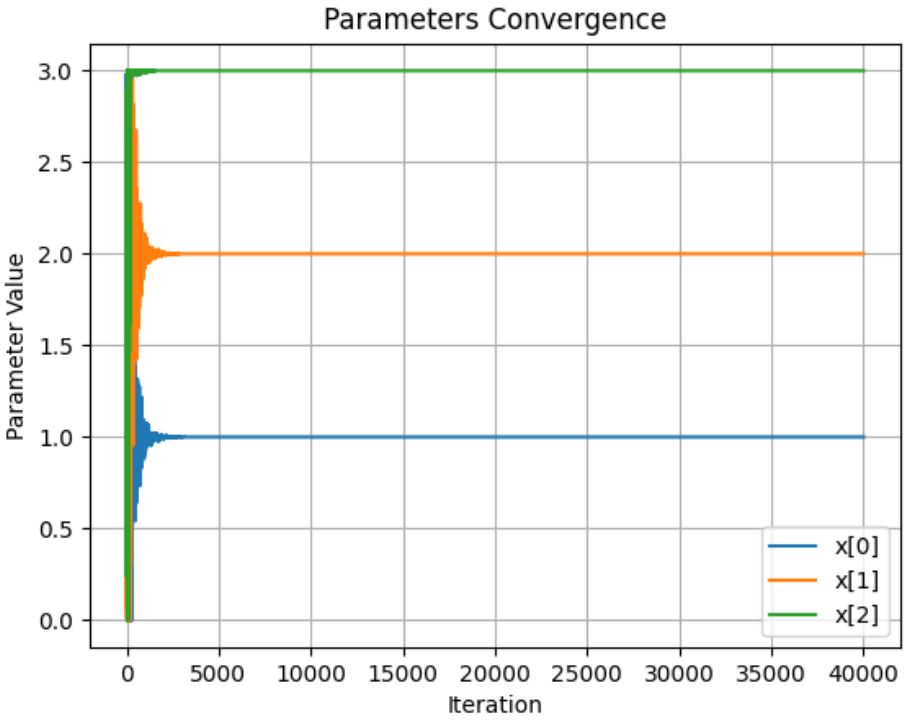

Fig. 3: Teste da convergência dos parâmetros

2.2 Otimização do controlador do robô seguidor de linha

2.2.1 Histórico de Otimização

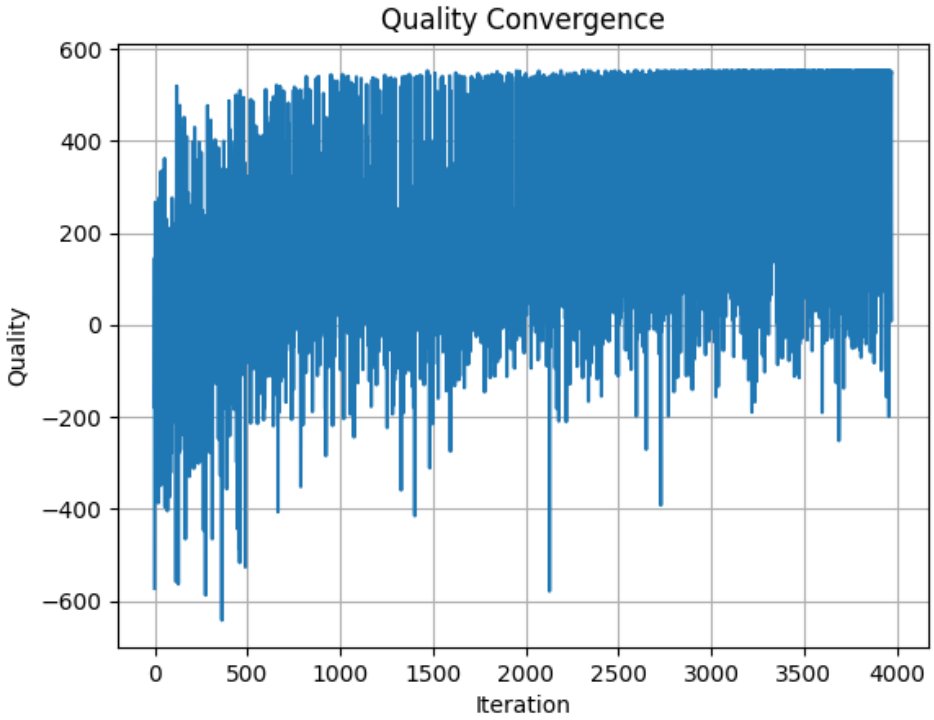

Fig. 4: Gráfico da Qualidade da Convergência

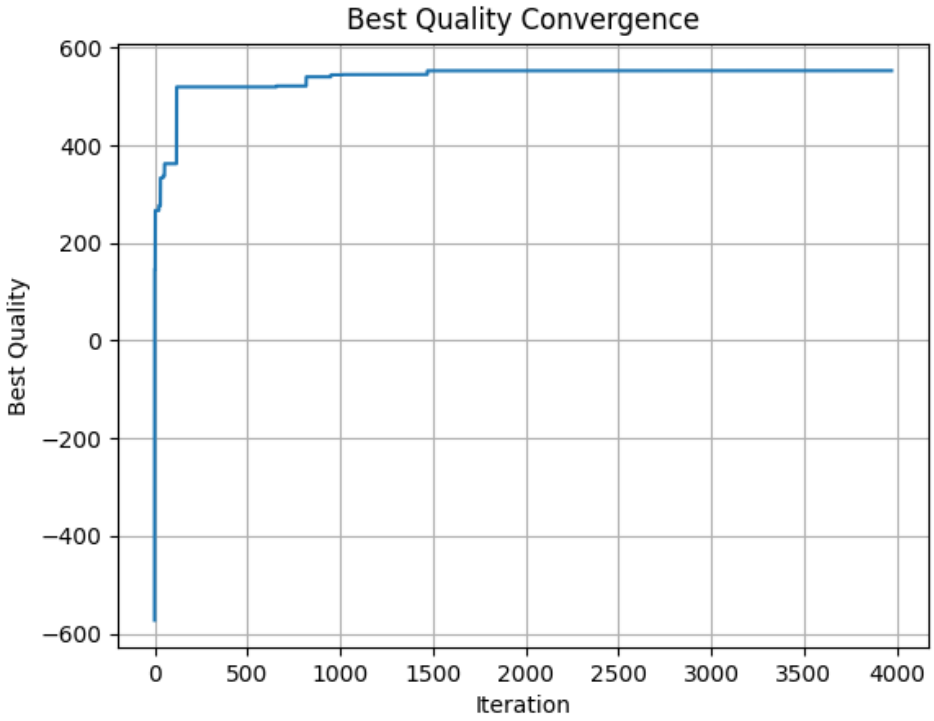

Fig. 5: Gráfico da melhor convergência

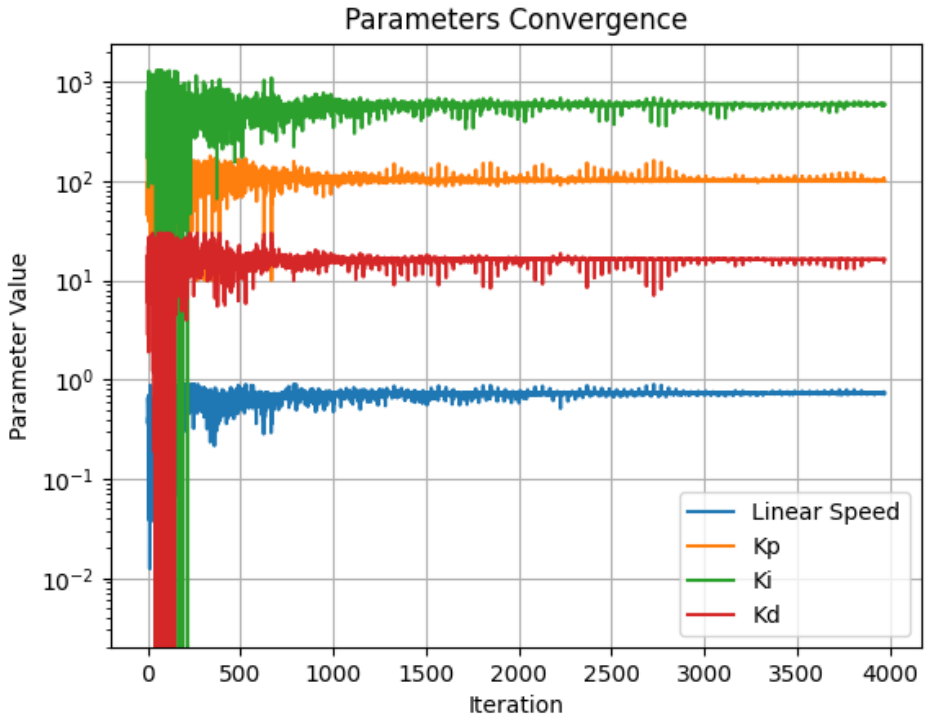

Fig. 6: Gráfico da convergência dos parâmetros do robô Seguidor de Linha

2.2.2 Melhor Trajetória Obtida Durante a Otimização

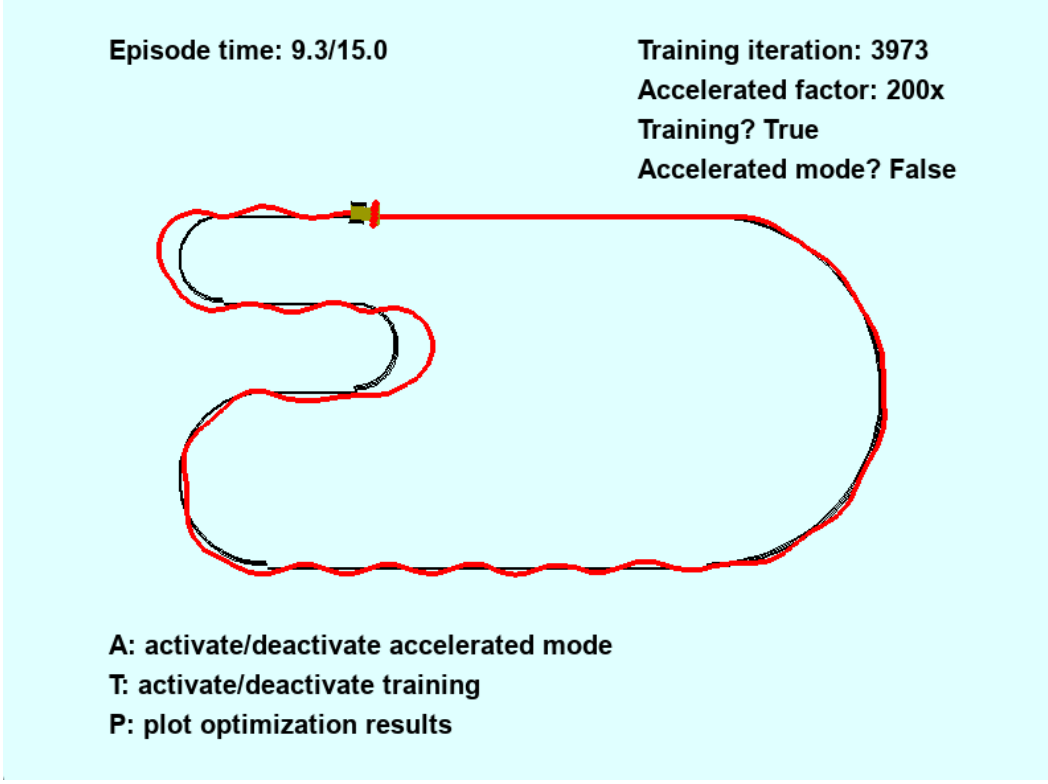

Fig. 7: Gráfico da melhor trajetória executada pelo Robô

3 Discussão sobre o observado durante o processo de otimização

Durante a execução do PSO para otimizar os parâmetros do controlador PID do robô seguidor de linha, foi possível observar uma evolução significativa na qualidade das soluções geradas ao longo das gerações. Nos primeiros episódios, o comportamento do robô era errático, com movimentações instáveis e, por vezes, incapacidade de seguir corretamente a trajetória estipulada. No entanto, à medida que o algoritmo ajustava os parâmetros, o robô passou a apresentar uma performance mais consistente, conseguindo se manter próximo à linha com menor oscilação lateral e maior eficiência de deslocamento. Todavia, devido ao caráter aleatório de parâmetros como r_p e r_g, ainda se percebia esse caráter errático em algumas partículas ao longo do tempo, mesmo quando havia um volume bem grande de iterações..

Os gráficos de convergência evidenciaram uma clara tendência de estabilização nos valores dos parâmetros ao longo das iterações. A recompensa acumulada por episódio apresentou um crescimento progressivo até atingir um platô, indicando que o algoritmo conseguiu escapar de regiões de baixa qualidade no espaço de busca e convergir para uma solução satisfatória — possivelmente um ótimo global.

A função de recompensa utilizada, desenhada para penalizar desvios em relação à linha e movimentos contrários ao sentido da pista, teve papel fundamental na orientação do processo

de otimização. Além disso, o valor adotado para o parâmetro de ponderação $w = 0,5$ revelou-se adequado para equilibrar a precisão no seguimento da linha com a rapidez na conclusão do percurso, contribuindo para um comportamento geral mais eficaz do robô.

Por fim, observou-se que, se o parâmetro definido para o erro, caso a linha não fosse detectada, fosse muito próximo a 0.03, o algoritmo tendia a "ignorar"esse fato, contentando-se com a recompensa obtida, de modo que, nas primeiras tentativas de otimização, o robô "travava"em soluções não ótimas. Ajustou-se, então, para valores em cerca de 1.0, o que conseguiu corrigir a trajetória durante o aprendizado.