INSTITUTO TECNOLÓGICO DE AERONÁUTICA

CT-213

RELATÓRIO DO LABORATÓRIO 04

TURMA 23-3

JOÃO LUÍS ROCHA SARMENTO

SÃO JOSÉ DOS CAMPOS

2020

IMPLEMENTAÇÃO DO PSO

O método PSO é um método de otimização que se baseia em população. Para esse laboratório, o objetivo da utilização do método era otimizar parâmetros de um robô seguidor de linha. No código, foi utilizado como hiperparâmetros: número de partículas da população (40), peso inercial (0.5), parâmetro cognitivo (0.6) e parâmetro social (0.8). Vale ressaltar, ainda, que, inicialmente, o peso inercial era 0.7, mas foi alterado para 0.5 após testes na simulação.

Além disso, antes de se de fato otimizar os parâmetros do controlador PID do robô, foi necessário testar a implementação do algoritmo PSO, para evitar despedício de tempo. Para isso, encontrou-se o máximo de várias funções relativamente simples (que podem ser analisadas analiticamente) no arquivo *"test_pso.py"*. Além da função previamente fornecida cujo máximo global é $[1, 2, 3]^T$ testou outras funções simples, com máximos parecidos (alterando os valores constantes de cada quadrado).

Para implementação do PSO, fez-se uso da divisão de métodos da classe ParticleSwarmOptimization previamente fornecida. Segue a divisão de métodos:

Método chamado ao se inicializar a classe:

- Nesse método (__init__), apenas inicializou-se as partículas utilizando os limites fornecidos. Isto é, x_i = random.uniform(lower_bound, upper_bound) e v_i = random.uniform(-delta, delta), em que delta = upper_bound - lower_bound. Além disso, criou-se parâmetros para se armazenar os melhores globais (posição e valor referente a posição) e, também, para armazenar o número da partícula que está sendo analisada.
- Vale ressaltar que a classe Partícula apenas contém a posição, velocidade e o melhor atual (posição e valor).

Método get_best_position:

- Esse método apenas retorna a posição que corresponde máximo global encontrado até o momento.

Método get_best_value:

- Esse método apenas retorna o máximo global encontrado até o momento.

Método get_position_to_evaluate:

- Esse método apenas retorna a posição da partícula que está sendo analisada.

Método advance_generation:

- Esse método é chamado caso uma geração completa seja analisada. Aqui, itera-se entre todas as partículas atualizando sua velocidade e posição.
- Para atualização da velocidade, utiliza-se a equação fornecida em aula: $v_i = w.v_i + \varphi_p r_p(b_p - x_i) + \varphi_g r_g(b_g - x_i)$, em que rp e rg são número aleatórios entre 0 e 1 gerados para toda partícula e os outros são parâmetros já calculados.
- Após isso, checa-se se a velocidade calculada não ultrapassou os limites iniciais (-delta e delta), e, caso tenha ultrapassado, coloca-se a velocidade como o limite mais próximo
- Da mesma forma, para se atualizar a posição utiliza-se a equação fornecida em aula: $x_i = x_i + v_i$.

- Após isso, checa-se se a posição calculada não ultrapassou os limites iniciais (lower_bound e upper_bound) e, caso tenha ultrapassado, coloca-se essa posição da partícula como o limite mais próximo.

Método notify_evaluation:

- Esse método atualiza os máximos da partícula e globais, e checa-se já se analisou todas as partículas.
- Caso se tenha analisado todas as partículas, reinicia-se a contagem e avança a geração (método advance_generation).

TESTE DE IMPLEMENTAÇÃO

Para executar o teste de implementação, utilizou a função:

$$f(x_1, x_2, x_3) = -((x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2)$$

Cujo máximo é facilmente obtido como sendo 0 com as variáveis (1,2,3). Observe os gráficos gerados durante o teste:

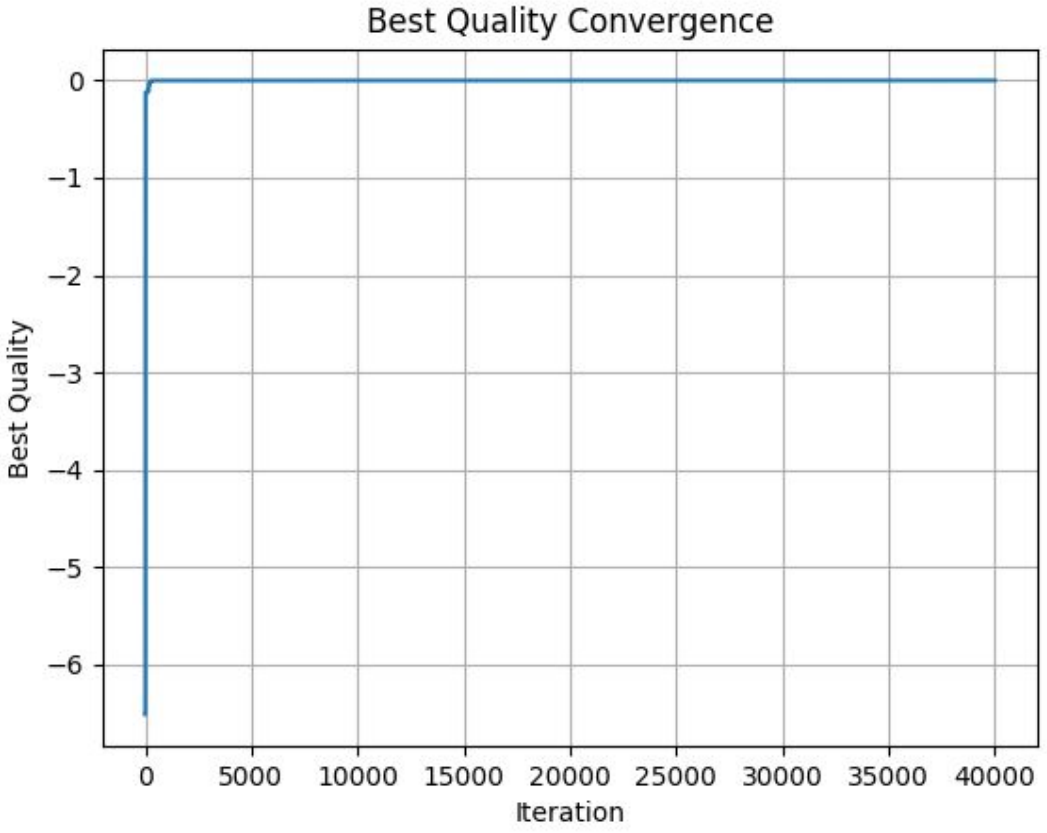
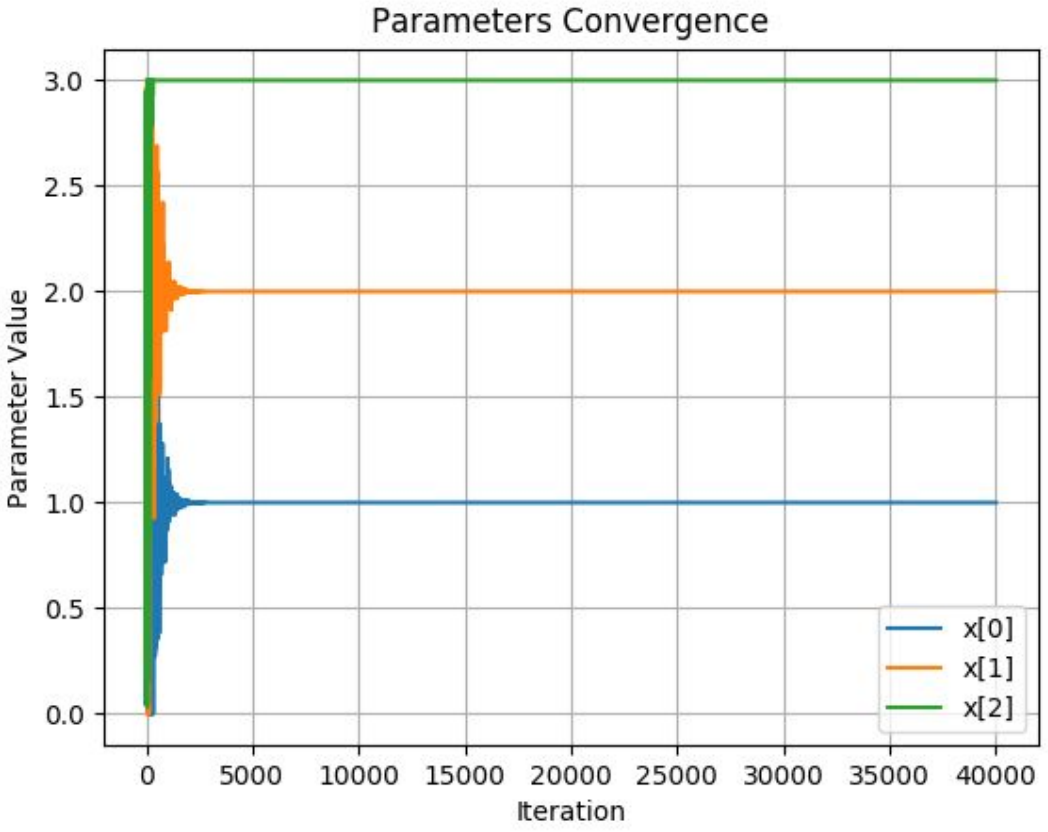


Figura 1.



Conforme esperado, ao se realizar o teste, se obteve como best_position (1,2,3) e como best_value (0).

OTIMIZAÇÃO DO LINE FOLLOWER

Para se realizar a otimização do robô utilizou uma simulação acelerada, para se obter os valores de velocidade linear e de kp, kd e ki do controlador PID. Para isso, pontuava-se o robô diante de algumas circunstâncias, buscando que ele fizesse o percurso da melhor forma possível. Os valores que fornecessem a maior soma de recompensa eram os valores armazenados para geração dos gráficos. Para implementação do sistema de recompensas, considerou-se a seguinte equação para uma recompensa k:

$$r_k = v_k.(< a,\ b >)\ - w.|e|$$

Nessa equação, o valor “vk” representa a velocidade linear, o vetor “a” representa a direção do robô, o vetor “b” representa a tangente à pista no instante, “w” é um peso para tornar o erro significante (esse peso foi colocado como 5.0 (valor alto para se ter certeza do percurso obtido) e “e” é o erro em relação a linha. Além disso, caso não seja detectado a linha, o erro é colocado como um valor padrão alto, para penalizar fortemente o robô. Neste código, foi considerado esse valor padrão como sendo 0.1 (maior que o máximo que seria 0.03). A partir daí, após 3460 iterações, pode-se obter os seguintes gráficos, trajetória e tabela de valores:

Kp	Ki	Kd	Velocidade linear	Recompensa total
136.96	549.64	13.28	0.59	455.53

Tabela 1. Valores obtidos

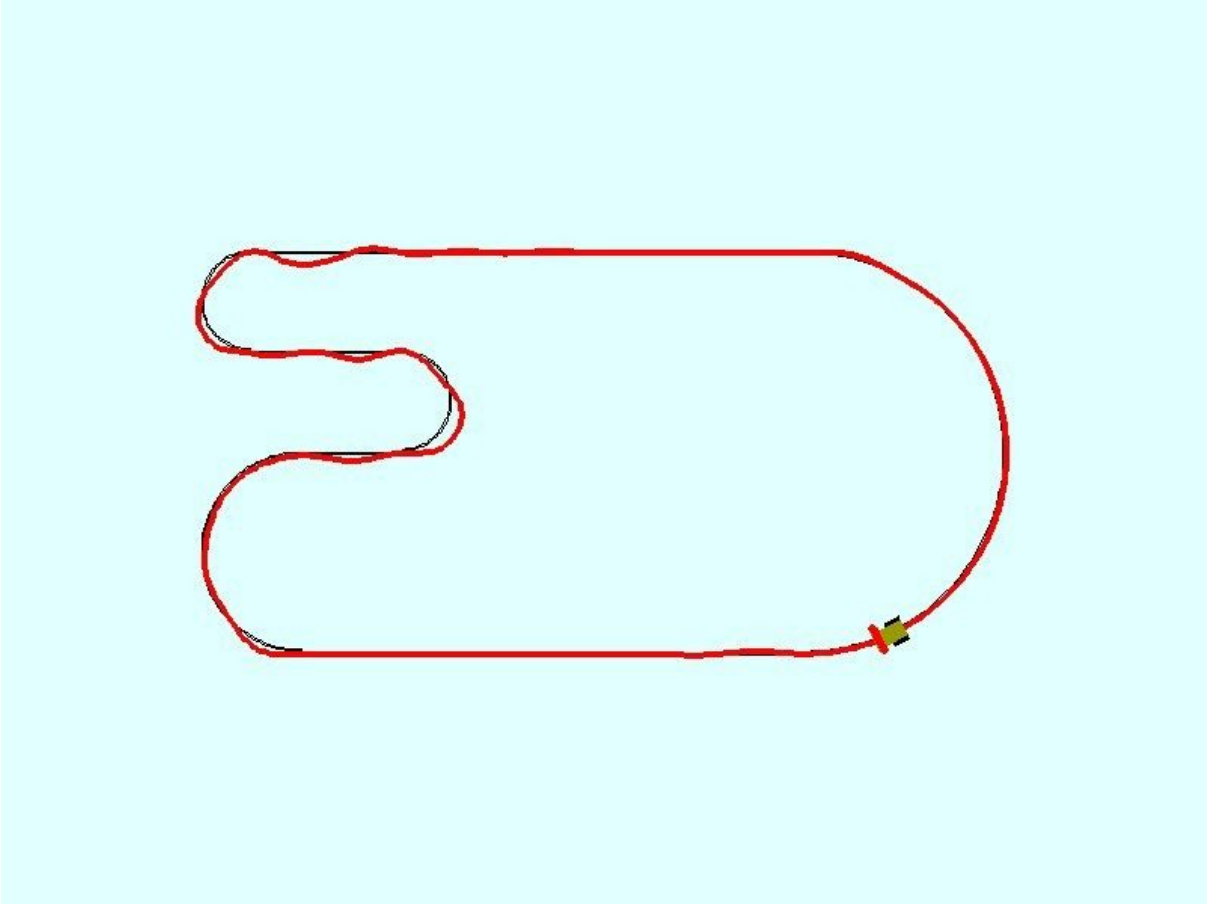


Figura 4. Trajetória obtida

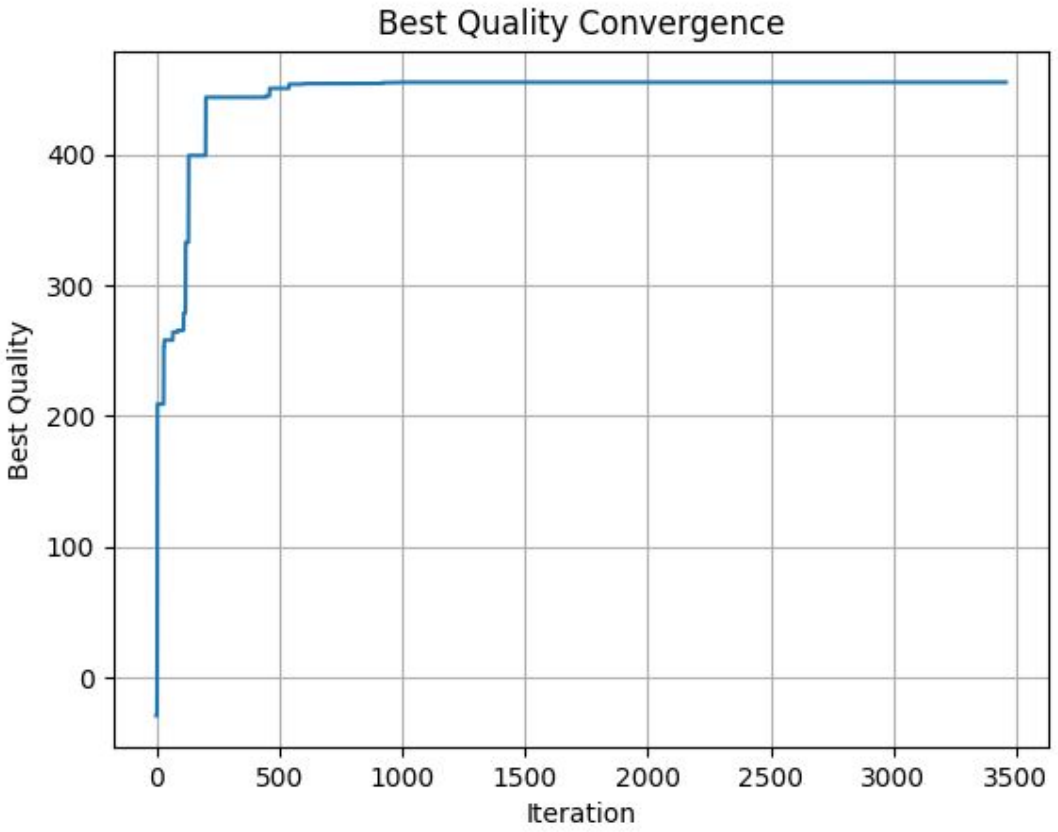


Figura 5.

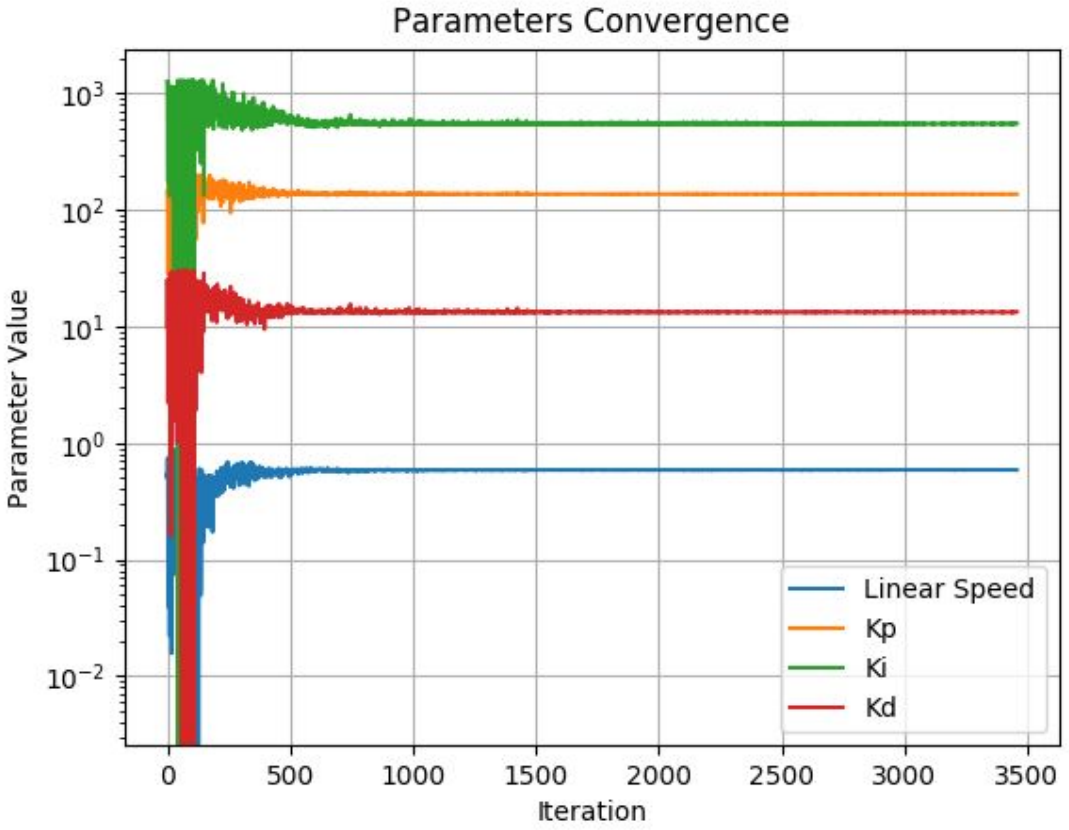

Parameters Convergence
Parameter Value
Iteration
Linear Speed
Kp
Ki
Kd
10^3
10^2
10^1
10^0
10^{-1}
10^{-2}
0
500
1000
1500
2000
2500
3000
3500

Figura 6.

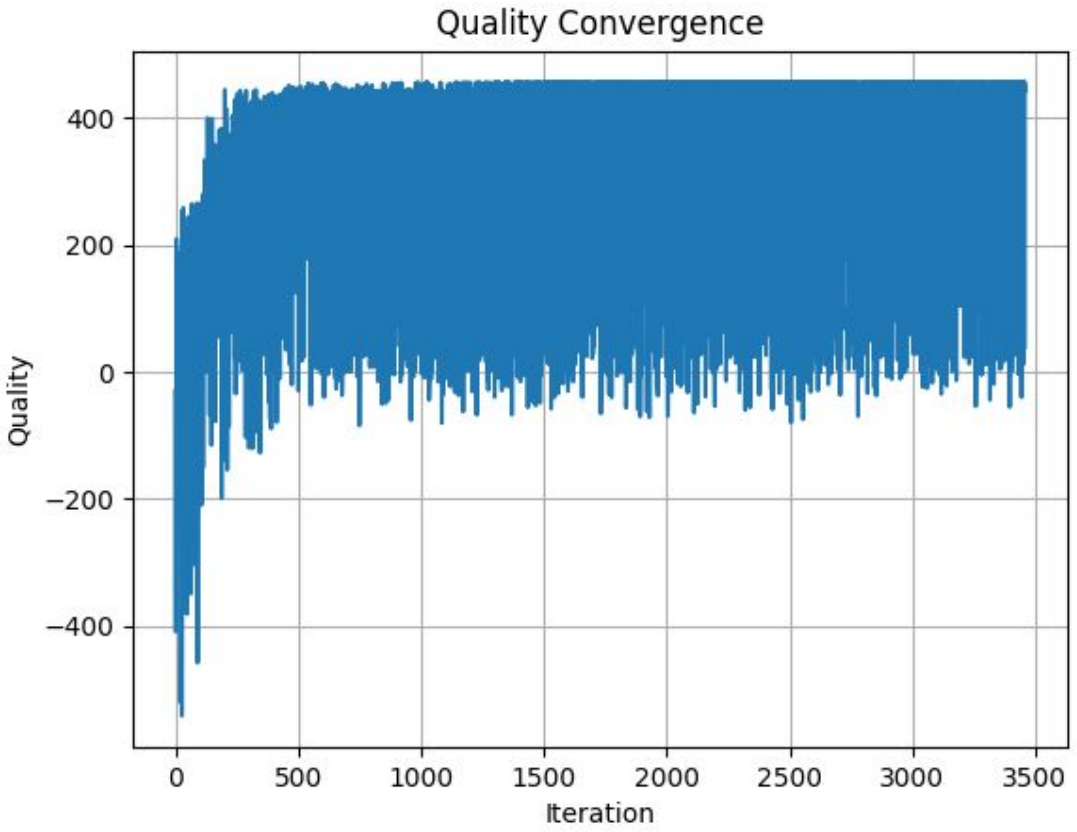


Figura 7.

Figura 2.

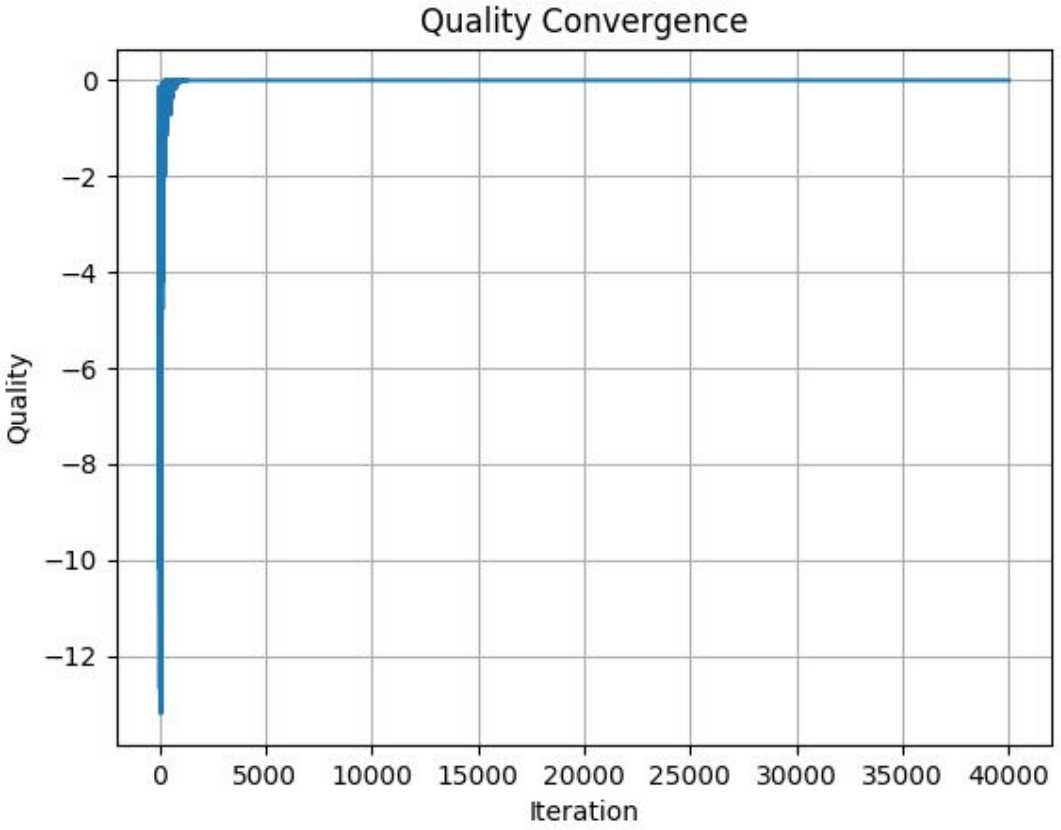


Figura 3.