IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 13, 2021, accepted May 21, 2021, date of publication May 25, 2021, date of current version June 4, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3083705

New PID Parameter Autotuning for Nonlinear Systems Based on a Modified Monkey–Multiagent DRL Algorithm

HONGMING ZHANG[1], WUDHICHAI ASSAWINCHAICHOTE[1], AND YAN SHI[2]

[1]Department of Electronic and Telecommunication Engineering, Faculty of Engineering, King Mongkut's University of Technology Thonburi, Bangkok 10140, Thailand

[2]Graduate School of Science and Technology, Tokai University, Kumamoto 862-8652, Japan

Corresponding author: Wudhichai Assawinchaichote (wudhichai.asa@kmutt.ac.th)

This work was supported in part by the Petchra Pra Jom Klao Scholarship, and in part by the Department of Electronic and Telecommunication Engineering, Faculty of Engineering, King Mongkut's University of Technology Thonburi.

ABSTRACT Proportional–integral–derivative (PID) control is the most widely used control law in industrial processes. Although various new controllers continue to emerge, PID controllers are still in a dominant position due to their simple structure, easy implementation, and good robustness. In the design and application of PID controllers, one of the core issues is parameter tuning. Accurately and effectively selecting the best tuning parameters of the PID is the key to achieving an effective PID controller. Therefore, this paper proposes a novel modified monkey-multiagent DRL (MM-MADRL) algorithm and uses it to tune PID parameters to improve the stability and performance of automatic parameter optimization. The MM-MADRL algorithm is a new version of the basic monkey group algorithm (MA) and the multiagent reinforcement learning algorithm known as the multiagent deep deterministic policy gradient (MADDPG). This paper selects a typical nonlinear quadcopter system for simulation; its principle and data are given below. MM-MADRL, the genetic algorithm (GA), particle swarm optimization (PSO), the sparse search algorithm (SSA) and differential evolution (DE) are used to adjust the parameters. The simulation results show that the overall performance of the MM-MADRL algorithm is better than that of the other algorithms.

INDEX TERMS PID controller, monkey algorithm (MA), multiagent deep deterministic policy gradient (MADDPG), modified monkey–multiagent DRL (MM-MADRL) algorithm, optimization.

I. INTRODUCTION

At present, the PID controller is one of the most widely used industrial controllers because it offers good adaptability, strong robustness, and a simple and clear algorithm structure. The literature [2] explains the basic principle structure of PID. Based on the advantages of PID control, PID controllers have been used for temperature control, switching power supply control, and control over air suspension, electronic throttles, quadcopters and the servomotor system. Studies [1], [3]–[5] and [37]–[40] introduced the application of PID control to different systems. Because PID control is widely used, improving PID control brings great convenience to our lives. The essence of the PID controller is to set and adjust its PID parameters to meet the various preset control performances. As early as 1942, Ziegler and Nichols proposed the Z-N method of PID regulator parameter tuning for an object of first-order inertia plus pure delay. Subsequently, different people proposed PID parameter tuning methods, but based on various control conditions, many controlled processes have characteristics such as a complex mechanism, a high degree of nonlinearity, a pure time lag, and time-varying uncertainty. The use of traditional empirical methods to adjust parameters is time-consuming and laborious, and the adjustment effect is not ideal. Therefore, the method of intelligent PID parameter self-tuning was proposed and has gradually become the direction of development.

At present, a variety of intelligent PID parameter self-tuning methods have been proposed [15]–[24], such as ant colony optimization (ACO), the genetic algorithm (GA), particle swarm optimization (PSO), and differential evolution (DE). These methods have achieved good research results in tuning PID parameters. However, these intelligent algorithms have their own advantages and disadvantages. The key is that

The associate editor coordinating the review of this manuscript and approving it for publication was Fei Chen.

it easily falls into the local optimal solution, and the calculation time of the DE and GA algorithms is too long to address these shortcomings. Later, more researchers improved the basic intelligent algorithms [6]–[14], [25]–[31] and [59]–[61] by combining the advantages of different algorithms and proposed improved algorithms such as improved ant colony optimization (IACO), enhanced whale optimization algorithms (EWOA), complex-order PSO (CoPSO) and unequal limit cuckoo optimization algorithm (ULCOA). While solving some of the defects of the original algorithm, these algorithms further improved the performance of the control system. In fact, most of the improvements to the original algorithm are improvements to the search mode of the algorithm itself, such as adding a selection operator and changing the calculation formula. Alternatively, combining the advantages of other algorithms allows them to be improved and tested. However, the influence of the initial parameter setting of the swarm intelligence algorithm and the influence of the artificially fixed search mode on the algorithm still exist. The wide application of PID control determines that the optimization of parameters should be based on different systems, and different system environments require different parameter optimization effects. Therefore, there is still much research space for the automatic tuning of PID parameters, as discussed in the literature [62]–[64]. These articles not only propose new improvements but also explain the necessity of PID parameter tuning. In recent years, the reinforcement learning algorithm has given researchers new inspiration. This algorithm uses the interaction between the agent and the environment to select the search action, which greatly reduces the limitations of the artificially set search mode and makes it the individual's own. The action information comes from the search environment itself. Therefore, combining the reinforcement learning algorithm with the swarm intelligence algorithm to improve the swarm intelligence algorithm is a new research direction. Therefore, this paper proposes a hybrid swarm intelligence and reinforcement learning algorithm, combining the advantages of the two such that they complement each other.

The monkey group algorithm [32] is an intelligent optimization algorithm proposed by Zhao and Tang in 2008. The algorithm is composed of the ''climb'', ''watch-and-jump'', and ''somersault'' operations. The ''climb'' and ''watch-and-jump'' operations form the monkey group algorithm, while the core search process, i.e., the ''somersault'' operation, enables the algorithm to jump out of the local optimal solution and perform a precise search in the later stage of the algorithm. The monkey group algorithm has the advantage over other algorithms of fewer adjustment parameters. The largest difference between the monkey group algorithm and other intelligent algorithms is the somersault operation, so the research object of this article is the monkey group algorithm. MADDPG [36] is a multiagent reinforcement learning algorithm. It obtains the optimal strategy through learning. It can give the optimal action by using only local information during its application. It does not need to know the dynamic model of the environment and the special communication requirements. Algorithms can be used not only in a cooperative environment but also in a competitive environment. Compared with traditional reinforcement learning algorithms, the MADDPG has a higher search speed [33]–[36].

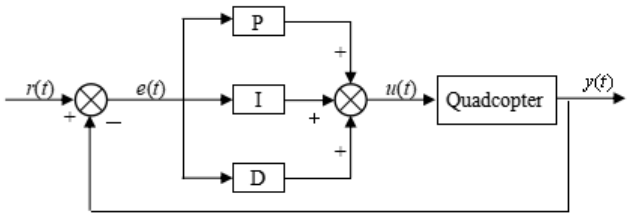


FIGURE 1. Basic principles of PID control.

Therefore, this paper introduces the framework of MADDPG into the core search part of the monkey group algorithm and solves the problem in which the initial step parameter setting of the monkey group algorithm has a greater impact on the results. Additionally, the monkey group algorithm itself is too independent of the monkey group individuals, and its iteration has a slight relationship with the environment. The MADDPG algorithm framework allows the monkey group individuals to carry out location information exchanges and is closely related to the changes in the search environment, increasing the search speed and accuracy. The ''somersault'' operation allows the algorithm to converge faster but also allows the algorithm to perform a precise search later. Based on the proposed MM-MADRL algorithm, many multiagent test environments have been proposed and tested. However, to optimize the parameters of the PID-controlled aircraft system with the proposed algorithm, the most important thing is to redesign and write a multiagent test environment, including the following: 1. An aircraft control system to be used in this article. 2. The optimization environment of the PID parameters to achieve the purpose of using the algorithm proposed in this paper to optimize the PID parameters. Additionally, the GA, PSO, and DE algorithms and the newly proposed (2020) SSA algorithm are used to compare the test results with the MM-MADRL proposed in this article. Therefore, the contribution of this article can be summarized as follows:

1. The MM-MADRL algorithm is proposed for self-tuning of the PID parameters.

2. The MM-MADRL algorithm improves the control performance.

3. An environment in which the MM-MADRL algorithm can adjust the PID parameters is designed.

4. Using Python 3.6 PyTorch == 1.5.1, for the code of MM-MADRL to tune the PID parameters, a simulation comparison with the GA, PSO, SSA and DE algorithms proves the superiority of the MM-MADRL algorithm.

II. PID CONTROLLER AND ALGORITHM INTRODUCTION

A. PID CONTROLLER

The principles of PID control are shown in Figure 1:

According to the deviation $e(t) = r(t) - y(t)$ between the given input value $r(t)$ and the actual output value $y(t)$, the PID controller divides it into proportional, integral and differential forms through a linear combination. The method constitutes the control quantity and then controls the controlled object. The control law is:

$$u(t) = K_P e(t) + K_I \int_o^\tau e(t)dt + K_D \frac{d}{dt} e(t) \tag{1}$$

where K_P is the proportional coefficient; K_I is the integral time constant; K_D is the derivative time constant; t is current time.

The PID control algorithm can be divided into two types in the application process, namely, the positional PID control algorithm and the incremental PID control algorithm. The two are the same in control theory, and each has its advantages and disadvantages.

This article uses the positional PID control algorithm in the simulation link, and its expression is as follows:

$$K_P e(k) + K_I T \sum_{j=0}^{k} e(j) + K_D \frac{e(k) - e(k-1)}{T} \tag{2}$$

where T is the sampling period, k is the sampling number, with $k = 1,2\ldots$, and $e(k-1)$ and $e(k)$ are the system deviation signals obtained at the $(k-1)^{th}$ and k^{th} moments, respectively.

B. MONKEY ALGORITHM (MA)

The monkey group algorithm is an intelligent algorithm that was recently proposed. It is different from other intelligent algorithms because of its ''somersault'' feature. Additionally, the monkey group algorithm has the advantages of fewer adjustment parameters and a simple structure, in contrast to other intelligent algorithms. Easy to operate, the ''somersault'' feature speeds up the optimization of the monkey colony algorithm, and it does not easily converge to the local optimal solution. The operation process of the monkey group algorithm is shown in Figure 2.

In Figure 2, the climbing process of the monkey group algorithm is shown. It is the process of monkeys performing local optimization in a small area. This process has two key parts: a pseudogradient calculation and a calculation of the fitness function value. The calculation of the pseudogradient provides the direction of movement for the monkey, and the calculation of the fitness function value completes the optimization operation, allowing a position with good fitness to replace a position with poor fitness.

The ''watch-and-jump'' operation is performed to ''wait and see'' around the current optimal position after the monkey individual completes the ''climb'' operation. If a better position is found, the current position is replaced. The key point in the operation is the calculation of the pseudogradient. The algorithm uses the pseudogradient of the objective function as the direction of movement. The pseudogradient calculation method is given below.

Suppose the pseudogradient of the point $x_{i,d}$ is $f_{i,d}$, where $\Delta x_i = \Delta x_{i,1}, \ldots., \Delta x_{i,d}$ is a random vector, θ is a random number, $\Delta x_{i,d} = a$, a is the set climbing step length, and d is the search range; then

$$f_{i,d}(x_{i,d}) = \frac{f(x_{i,d} + \Delta x_{i,d}) - f(x_{i,d} - \Delta x_{i,d})}{2\Delta x_{i,d}} \tag{3}$$

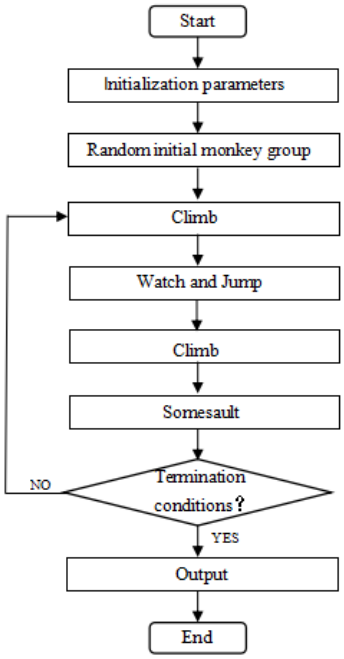


FIGURE 2. Monkey algorithm process [32].

The ''somersault'' operation is performed to make the monkey jump out of the local optimal position in the current area and continue to ''climb'' and ''watch-and-jump'' in a new range to find the global optimal solution.

The main parameters of the monkey group algorithm are as follows:

TABLE 1. MA parameters.

	Quantity
M	Total number of monkeys
MT	Maximum number of iterations
N_C	Maximum number of iterations of the "climbing" operation
a	"Climb" operation step parameters
b	"Watch-and-jump" operation field of view length
c	Upper bound of the "somersault" interval
d	Lower bound of the "somersault" interval

The pseudocode of the monkey group algorithm is as follows:

In the pseudocode above, n is the dimension of the problem, x is the monkey, y is the new monkey, and x' is the current monkey.

According to the literature, the monkey swarm algorithm has the following shortcomings:

1. It has only a single initialization.

Algorithm 1 Monkey Algorithm (MA)

Initialization():
for $i = 1: M$ do
  for $j = 1: n$ do
    $x(i, j) \leftarrow$ random
  end for
  fitness $\leftarrow f(x(i, :))$
end for
iteration $\leftarrow 1$
While *iter* $< MT$ do
  Climb():
    $k \leftarrow 0$
  for $i = 1: M$ do
  While $(k < N_C)$ do
    for $j = 1: n$ do
      $p \leftarrow rand(1)$
      if $p > 0.5$ then $x \leftarrow a$
      else $x \leftarrow -a$
    Check whether it is out of range; if
      it is out of range, it is equal to the critical value
      end for
        $ch \leftarrow$Difference between critical values
  for $i = 1: M$ do
    $f' \leftarrow \frac{\text{ch}}{2\Delta X}$
    $y \leftarrow x - a\sin(f'(x'))$
  end for
  if y in the feasible area
  $x' \leftarrow y$
  $k \leftarrow k + 1$
    end while
    end for
Watch-and-Jump():
for $i = 1: M$ do
  While (***True***)
    for $j = 1: n$ do
      $l \leftarrow x - bu \leftarrow x + b$
        $y \leftarrow 1 + rand(1)(u - 1)$
    end for
    if $f(y) \leftarrow f(x')$ jump to end
    end while
    end for
    do climb()
    somersault():
    Randomly take the number in the somersault interval
      (c, d) as the step length of the somersaul**t**
    if y is in the feasible region
        $x' \leftarrow y$
    else Return to the first step.
    end
    $iter \leftarrow iter + 1$
  end while

2. The fixed a and b cause the monkey's movement to be unable to interact with the environment, which affects the optimization ability.

3. There is no communication and cooperation between individual monkeys, which prevents the algorithm from adjusting individual monkey behaviors according to the situation, which affects the speed of optimization.

C. REINFORCEMENT LEARNING AND MULTIAGENT REINFORCEMENT LEARNING

Reinforcement learning tasks are usually described by the Markov decision process (MDP). The principle is shown in the figure below:

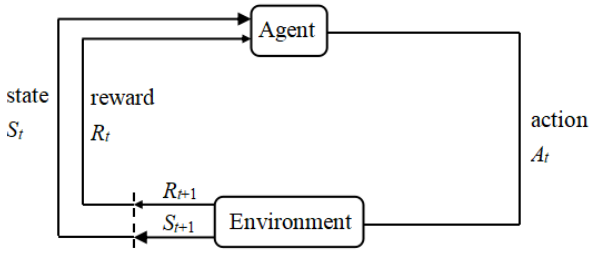


FIGURE 3. Fundamentals of reinforcement learning.

According to the above figure, when agent performs a certain task, it first interacts with the environment to generate a new state, and the environment gives a reward. The cycle continues, and the agent and the environment continue to interact to generate more new data. The reinforcement learning algorithm is used to interact with the environment through a series of action strategies to generate new data and then uses the new data to modify its own action strategy. After several iterations, the agent learns the action strategy needed to complete the task.

In summary, the key elements of reinforcement learning are as follows:

TABLE 2. Important parameters of reinforcement learning.

Symbol	Quantity
Agent	Mainly related to policy, value function and model
Reward	Based on the size of the reward given by the environment
Action	Agent action
State	Information contained in the agent's environment
Environment	Reinforcement learning environment

Q-learning is one of the traditional model-free reinforcement learning algorithms with which we are familiar. However, traditional reinforcement learning is limited to situations where the action space and sample space are very small and generally discrete. However, tasks that are more complex and closer to actual situations often have a large state space and continuous action space. Therefore, the researchers proposed deep learning that combines high-dimensional input with reinforcement learning, taking

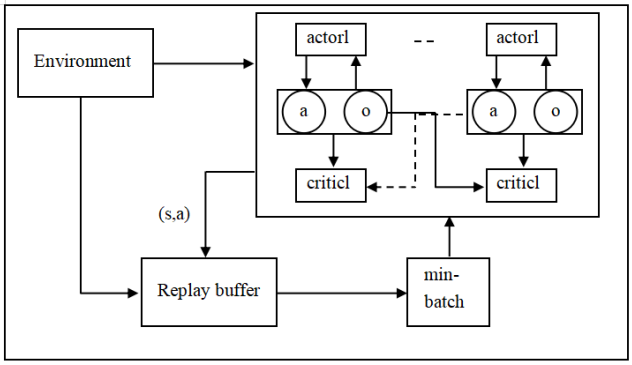


FIGURE 4. Fundamentals of the MADDPG.

advantage of both, and then performing deep reinforcement learning.

Deep Q-learning (DQN) and DDPG are well-known deep reinforcement learning algorithms. However, the above algorithms are suitable only for a single-agent environment. When addressing actual problems, multiagent scenarios are very common. Thus, a multiagent reinforcement learning algorithm is proposed, i.e., the multiagent deep deterministic policy gradient (MADDPG).

In recent years, reinforcement learning algorithms have been applied to many fields, and many researchers have studied various reinforcement learning algorithms and proposed improvements, as described in the literature [41]–[54]. A review of the literature shows that when deep reinforcement learning uses neural networks to fit traditional reinforcement learning, it has more advantages and offers wider applicability than traditional reinforcement learning. The proposal of multiagent deep reinforcement learning (MADRL) offers additional reinforcement. The application of learning algorithms has taken a step forward.

MADDPG is a series of improvements that have been made to the AC algorithm to make it suitable for complex multiagent scenarios that cannot be handled by traditional RL algorithms. Figure 4 shows the basic framework of the MADDPG. The schematic diagram and pseudocode of the MADDPG are given below.

The MADDPG is an algorithm based on a policy gradient. It can be traced back to the DPG algorithm. After the DPG algorithm, the neural network is used for fitting so that the DDPG algorithm can be used. The MADDPG algorithm extends the DDPG algorithm to many agents in the environment.

The calculation of its gradient is also clearly indicated in the code:

$$\nabla_{\theta_i} J \approx \frac{1}{S} \sum_i \nabla_{\theta_i} \boldsymbol{\mu}_i \left(O_i^j\right) \nabla_{a_i} Q_i^{\boldsymbol{\mu}} \times \left(\mathbf{x}^i, a_1^j, \ldots, a_i, \ldots, a_N^j\right) \Big|_{a_i = \boldsymbol{\mu}_i(o_i^j)} \quad (4)$$

The method of gradient ascent can ensure that each strategy is better than the previous strategy and that a single agent can approach the correct direction at each step under these conditions.

Algorithm 2 Multiagent Deep Deterministic Policy Gradient (MADDPG)

for episode = 1 to M do
Initialize a random process $\mathcal{N}$ for action exploration, receive initial state information x.
for t = 1 to max_episode_Length do
For each agent i, select the action $a_i = \boldsymbol{\mu}\theta_i\,(o_i) + \mathcal{N}_t$
Return to the collection of all the agents' actions a = $(a_1, \ldots a_N)$,
Store reward r and new state x' (x, a, r, x') in the relay buffer $\mathcal{D}$ $x \leftarrow x'$
for agent i = 1 to N do
Sample random min-batch of S samples $\left(\mathbf{x}^j, a^j, r^j, \mathbf{x}'^j\right)$ from $\mathcal{D}$
Set $y^i = r_i^j + \gamma Q_i^{\mu'}\left(\mathbf{x}'^j, a'_j, \ldots, a_N\right)\Big|_{a'_K = \mu'_k(o_k^j)}$
Update critic by minimizing the loss:
$\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q_i^{\mu}\left(\mathbf{x}^j, a_1^j, \ldots, a_i, \ldots, a_N^j\right)\right)^2\Big|_{a_i = \mu_i(o_i^j)}$
Update actor using the sampled policy gradient:
$\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i}\boldsymbol{\mu}_i\left(o_i^j\right)\nabla_{a_i} Q_i^{\boldsymbol{\mu}}$
$\left(\mathbf{x}^j, a_1^j, \ldots, a_i, \ldots, a_N^j\right)\Big|_{a_i = \boldsymbol{\mu}_i\left(o_i^j\right)}$
end for
Update target network parameters for each agent i
$\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$
end for
end for

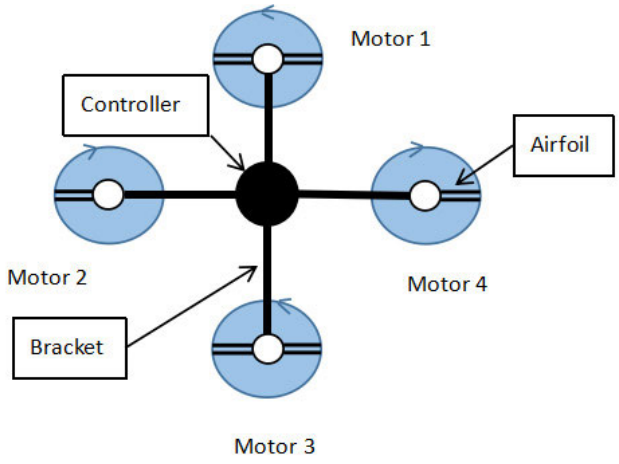


FIGURE 5. Quadcopter structure.

D. QUADCOPTER SYSTEM

The principal structure of the quadcopter system is given in Figure 5.

Viewed from the top, it has 2 clockwise rotating rotors and 2 counterclockwise rotating rotors. Its working principle is

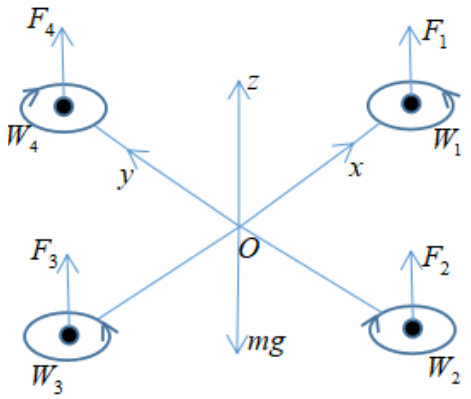


FIGURE 6. Principle of vertical motion.

based on the basic principles of fluid mechanics. The upper and lower surfaces of the rotor form a pressure difference to generate lift.

The state of motion in space is divided into the following 6 types:

1. Vertical motion.
2. Pitch motion.
3. Roll motion.
4. Yaw motion.
5. Back and forth motion.
6. Lateral motion.

The kinematic equations and principles of various motions can be understood in detail from the literature [58]. This article uses vertical motion in the simulation part, so the focus is on vertical motion.

The vertical motion schematic diagram is shown in Figure 6.

$F_{1,\ldots,4}$ is the lift of the indicated rotor, $W_{1,\ldots,4}$ is the motor speed, m is the quality and g is the acceleration of gravity.

Under the condition that the rotation speed of the four rotors is the same, when $F > mg$, the aircraft rises vertically; when $F < mg$, the aircraft descends vertically; and when $F = mg$, the aircraft is hovering.

The nonlinear mathematical model of the hovering state can be found in the literature [58], and this reference also provides detailed notes and the processes by which to derive the formulas.

III. MM-MADRL DESIGN PRINCIPLES AND IDEAS

A. THE BASIC IDEA OF MM-MADRL

First, the core search method of the monkey swarm algorithm is to perform the ''climb'' and ''watch-and-jump'' operations in space D to achieve the goal of optimization. Then, the search space D is regarded as the action strategy space of RL. The monkey group individuals in the space are regarded as agents, the fitness function value is regarded as the reward of the environment, and the search problem of the monkey group algorithm can be easily transformed. This is a multiagent reinforcement learning problem. Based on the MADDPG, each agent has an actor and a critic. The critic can obtain global information. Actors can take actions based only on local observations. We can also call actor networks. The policy network is responsible for making decisions for the agent, and its input is the state of the agent. The critic network is generally called a critic and is responsible for evaluating the value of the actor making this decision. Each individual monkey is regarded as an MADDPG agent and accordingly MADDPG creates N individual monkeys; then, a concurrent action of N monkey individuals is a search for space. Based on the essence of the monkey swarm algorithm optimization, multiple monkeys have a common goal in the search space D, which is to find the position with the best fitness function value, and the search environment is regarded as a multiagent cooperative environment.

Initialization:

The search space D is defined for segmentation $D = (D_1, \ldots..D_N)$, which is the environment. The total number of monkeys, M, is the number of agents. Each individual monkey appears in a different small area of space D. The monkey's movement mode n is defined, which is the dimension of the problem, denoted as $\overrightarrow{a} = (\overrightarrow{a_1}, \ldots \overrightarrow{a_N})$. Then, the fitness function is set, which is the reward r. Finally, the monkey position is initialized randomly, denoted as x.

Core section:

(1) The agent perceives the state x of the environment at time t.

(2) According to the current state x and the enhanced greedy information, the system selects and then executes a certain action port $\overrightarrow{a}$, the action port $\overrightarrow{a}$ acting on the current environment, and the environment changes accordingly.

(3) The current optimal position $best\_x'$ is recorded.

(4) The reward function is received by the agent, and the existing greedy strategy changes, that is, $t - t + 1$.

(5) When the area r is stable, the ''somersault'' operation is performed to make the agent jump into an area closer to the target and continue with steps (1)-(4).

(6) When a satisfactory target state is obtained, the cycle stops and outputs the optimal position, $Best\_x$.

The core search method of this operation allows each individual monkey to act based on the environment. Additionally, the step length parameter and the field of view parameters a, b cannot be set without the need to specifically set the step length and field of view parameters so that the monkey individual random selection is within a given range, because based on the greedy strategy, the monkey's ''climbing'' and ''watch-and-jump'' operations are performed at the same time, which ensures the search diversity and improves the accuracy. In the original ''climbing'' operation with a pseudogradient as the search direction, after the change, based on the environment, the direction of good adaptability is the new search direction. The essence of optimization has not changed, and because of the advantages of the MADDPG, the cooperation between monkeys increases, and the search becomes faster.

Algorithm 3 Modified Monkey – Multiagent Deep Reinforcement Learning Algorithm (MM-MADRL)

Set cycle search times MT
for episode = 1 to MT do
Initialize a random process $\Im$ for action exploration,
N_C receive initial state information x.
Set the number of search rounds episode_length=
split search space $D = (D_1, \ldots D_N)$
for $t = 1$ to max_episode do
For each monkey j, select the action $a_j = u\theta_j(o_j) + \Im(t)$
Return to the collection of all the monkeys' actions $\overrightarrow{a} = (\overrightarrow{a_1}, \ldots \overrightarrow{a_N})$
Store reward $r = (r_1, \ldots .r_N)$ and new state X' $(x, \overrightarrow{a}, r, x')$ in relay buffer $\mathcal{D}$
Record the best position $best\_x'$ for each episode_length
Set the stop condition of the loop episode >10 && r'-r < 0.001
do somersault ()
Start interval update after NC rounds from $best\_x'$ to obtain (MAX_x, MIN_x),
$c \leftarrow x - (x - MAX\_x)0.1 \quad d \leftarrow x - (x - MIN\_x)0.1$
Return to the main loop
Reach the set goal
Output the global $Best\_x$
for monkey i = 1 to N do
Sample random min-batch of S samples (X^j, a^j, r^j, X'^j) from $\mathcal{D}$.
Set $y^j = r^j + \gamma Q_j^{u'}(X'^j, a'_1, \ldots, a'_N)\,|_{a'_k = u'_k(o_k^j)}$
Update critic by minimizing the loss:
$\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q_i^{\mu}\left(\mathbf{x}^j, a_1^j, \ldots, a_i, \ldots, a_N^j\right)\right)^2 \Big|_{a_i = \mu_i(o_i^j)}$
Update actor using the sampled policy gradient:
$\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i}\boldsymbol{\mu}_i\left(o_i^j\right)\nabla_{a_i} Q_i^{\boldsymbol{\mu}}$
$\left(\mathbf{x}^j, a_1^j, \ldots, a_i, \ldots, a_N^j\right)\Big|_{a_i = \boldsymbol{\mu}_i\left(o_i^j\right)}$
end for
Update target network parameters for each monkey i
$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$
end for
end for

"Somersault" step:

The $best\_x'$ based on the core search algorithm updates the search area to form a new search area (c, d), allowing each individual monkey to jump to the new search area to continue the search, that is, an area closer to the target location for further search, which accelerates the search convergence. Additionally, it also ensures a precise search within a small range in the later stage and improves the accuracy.

The pseudocode of MM-MADRL is as follows:

B. MM-MADRL ALGORITHM ANALYSIS

Multiagent cooperation environment:

The stochastic game (SG), or Markov game, can be defined as:

$$\langle A, X, \{U_i\}_{i\in A}, f, \{p_i\}_{i\in A}\rangle \tag{5}$$

where A represents the number of agents, X is the state space, $\{U_i\}_{i\in A}$ is the action space, U is the set of multiagent joint actions, the transition probability distribution p_i: X^*U^*X, and $i \in A$ is the reward function.

Definition 1: If X = empty set, then the Markov game is a static game; when $p_1 = p_2 = \ldots\ldots .p_N$, the agent has a completely cooperative relationship.

According to *Definition* 1, the learning objective is simplified to the MDP, and the form of Q in the joint action space is:

$$Q_{k+1}(x_k, u_k) = Q_k(x_k, u_k) + a[r_{k+1} + \gamma \max_{u'} Q_{k+1}(x_{k+1}.u') - Q_k(x_k, u_k)] \tag{6}$$

In the cooperative environment, agents can be divided into direct collaboration and indirect collaboration without collaboration.

Strategy gradient and gradient ascent:

As mentioned above, the MADDPG algorithm is an algorithm based on a policy gradient. Simply put, its purpose is to find an optimal strategy for the agent to obtain the maximum reward. Its essence lies in directly modeling and optimizing the strategy. The homogeneous modeling method is usually a θ parameterized function $\pi_\theta(a|s)$, and its reward function is defined as:

$$J(\theta) = \sum_{s\in S} d^{\pi}(s)V^{\pi}(s) = \sum_{s\in S} d^{\pi}(s)\sum_{a\in A}\pi_\theta(a\mid s)Q^{\pi}(s,a) \tag{7}$$

where $d^{\pi}(s)$ is the stationary distribution of the Markov chain derived from π_θ, $V^{\pi}(s)$ is the expected cumulative return of the state following strategy π, $s \in S$ is the state, $a \in A$ is the action, and $Q^{\pi}(s, a)$ is the action-value function following strategy π.

Gradient ascent ensures that every new strategy is better than the old strategy. θ changes in the direction of $\nabla J(\theta)$.

IV. PID PARAMETER TUNING BY THE MM-MADRL ALGORITHM

According to MADRL, analyze the three parameters of PID control, i.e., k_p, k_i and k_d, and build a search environment.

Based on [33], an environment with three parameters can be built, as shown in Figure 7.

The cuboid environment (Figure 5) is divided equally into 8 small cubes.

Assign an agent to each small cube, with the initial position of each agent being random. Therefore, the number of agents designed in this article is 8. Each agent is a set (k_p, k_i, k_d). Thus, the motion of each agent is a three-dimensional motion. The movement of each agent can be recorded in detail as follows:

The second step is to set the reward function. In the PID control system, the evaluation of the merits of the PID parameter values can usually be measured by the deviation integral

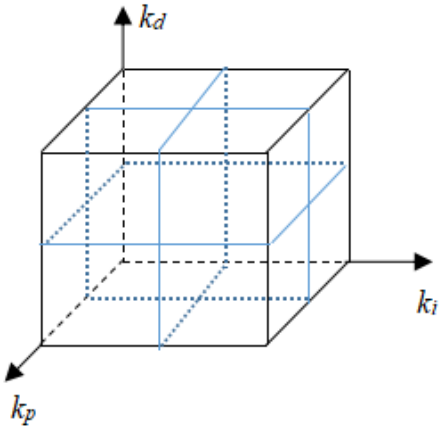


FIGURE 7. Environment of three parameters.

TABLE 3. Example of agent movement.

	Increase	Constant	Reduce
k_p	√		
k_i		√	
k_d		√	

By analogy, some parameters do not move while the other 1-2 move, all 3 move, or all 3 do not move.

index. There are three commonly used deviation integral indicators:

$$IE = \int_o^\infty e(t)\,dt$$
$$ISE = \int_o^\infty e^2(t)\,dt$$
$$IAE = \int_o^\infty |e(t)|dt \tag{8}$$

Different index selections have different results in the optimization of the system. In the actual operation process, you can choose the index function according to the purpose of your own system. Some researchers have also proposed an index function combining multiple indexes. This article uses the *ISE* indicator function to design the fitness function.

Thus, the reward can be set as:

$$R = \begin{cases} \text{Good adaptability}, & r = \text{Positive reward} \\ \text{Bad adaptability}, & r = \text{Negative reward} \\ \text{Achieve goals}, & r = \text{Large positive reward} \end{cases}$$

Set a loop search with the following loop stop condition: loop a certain number of times, after which the reward stabilizes. The two loop training rewards stops if the change is small.

This paper designs a multiagent deep reinforcement learning environment. Simply put, the three parameters k_p, k_i and k_d are randomly combined within a specified range, and the pros and cons of the combinations are judged by r. For the optimal combination value, if there is a definite objective function, it can be used as the judgment function; if there is no definite objective function, such as in a nonlinear system, the above three index functions can be used for judgment.

TABLE 4. Target value test result.

Function	DE_{BEST}	PSO_{BEST}	GA_{BEST}	SSA_{BEST}	MM-MADRL $_{BEST}$
Ackley	4.44089 21e-16	1.215472 17e-12	1.5258796 9e-07	4.440892 1e-16	0
Griewank	0	0	0.0296	0	0
Rastrigin	0	0	0	0	0

TABLE 5. Average target value test result.

Function	DE_{AVE}	PSO_{AVE}	GA_{AVE}	SSA_{AVE}	MM-MADRL $_{AVE}$
Ackley	4.44089 21e-16	1.2296830 2e-12	1.5258796 9e-07	4.44089 21e-16	0
Griewank	0	0.007396	0.04683	0	0
Rastrigin	0	0	0.99	0	0

V. SIMULATION AND RESULTS ANALYSIS

The simulation part of this article uses Python 3.6 and PyTorch == 1.5.1 to write the code of MM-MADRL, GA, PSO, DE, and SSA and to write the PID control environment code of a quadcopter at the same time. The following details the steps of the simulation and the analysis of the results obtained. Additionally, for simplicity, the Ornstein–Uhlenbeck process (OU) noise, which is used in deep reinforcement learning and is also correlated with the signal, is applied in the simulation part in this article. The simulation is carried out using Intel(R) Core(TM) i7-6700T CPU @ 2.80GHz with 64-bit operating system, Windows 10, and Pycharm 2020.3.3 Professional Edition, while the output calculation time is through the system Direct statistics.

A. TEST FUNCTION SIMULATION

To test the performance of MM-MADRL, this paper selects 3 standard test functions as the fitness function of the algorithm for testing. These functions have different mathematical characteristics, such as single peaks and multiple peaks. The functions tested in this paper include the Ackley function, Rastrigin function and Griewank function. Additionally, the GA, PSO, SSA and DE algorithms are brought into and tested in the same environment. The maximum number of iterations is 1000, the total group number $pop = 50$, and $dim = 2$. Each algorithm was tested 10 times for each function, and the results obtained are as follows:

From Tables 4 and 5, the target value is 0 for the test functions. The tables show the best and average search results from DE, GA, SSA, PSO and MM-MADRL. The result of

TABLE 6. Operation time result.

Function	DE_{BEST} (sec)	PSO_{BEST} (sec)	GA_{BEST} (sec)	SSA_{BEST} (sec)	MM-MADRL $_{BEST}$ (sec)
Ackley	2.20	0.08	0.87	1.30	0.05
Griewank	0.65	0.37	1.02	1.38	0.28
Rastrigin	1.89	0.32	1.19	1.53	0.32

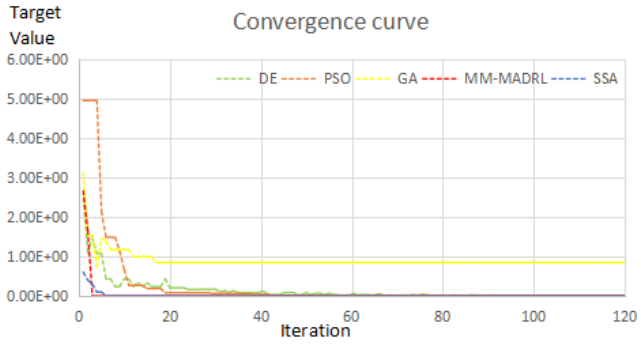


FIGURE 8. Griewank function convergence curve.

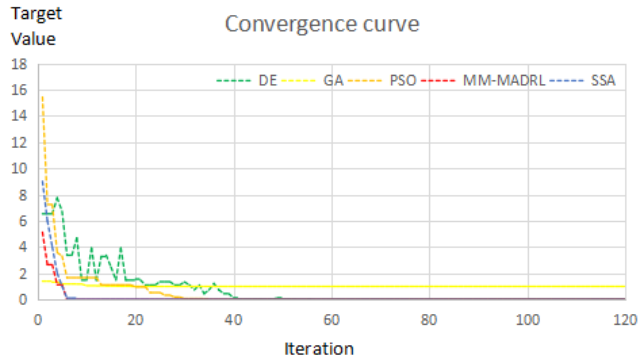


FIGURE 9. Rastrigin function convergence curve.

the MM-MADRL algorithm is better than that of the other algorithms. Table 6 illustrates the optimal time for several algorithms to find the target. The time in the table is the total time from the beginning to the end of the algorithm and outputting the target value. DE takes the longest time, and the best performance is the MM-MADRL algorithm proposed in this paper.

In summary, the MM-MADRL algorithm performs best in the test function experiment and has advantages in calculation time and accuracy.

The convergence curve is shown in Figures 8-10.

Figures 8, 9 and 10 show the average convergence curves of the GA, DE, PSO, SSA and MM-MADRL algorithms in the three test functions. The MM-MADRL algorithm proposed in this paper can quickly find the target value and end.

B. QUADCOPTER SYSTEM SIMULATION

First, we introduce the quadcopter. The quadcopter is a typical nonlinear system. Based on the control structure of the

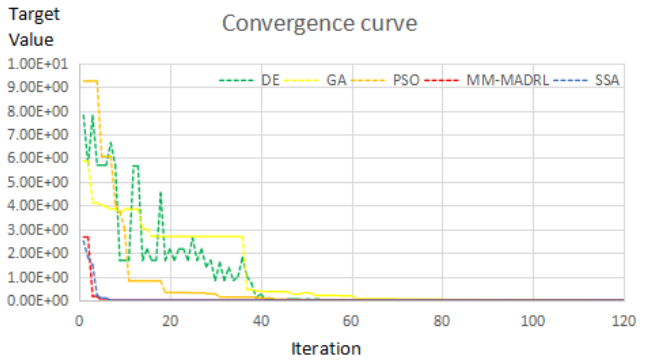


FIGURE 10. Ackley function convergence curve.

literature [58], the simulation part is designed. This article sets the initial altitude to 0 and requires the aircraft to fly to a height of $H = 10$ m and remain there. The mass of the aircraft $m = 1.54$ kg, the transmission constant of the electromechanical system $c = 10$, the known acceleration of gravity $g = 9.81$ N/kg, and the resistance is 0.75speed. This article compares the control force u output by the PID controller with that of the quadcopter. A cycle is set in the flight state to simulate the movement of a four-axis aircraft. Because the objective function of a nonlinear system is difficult to determine, due to Figure 6, the specific cycle settings are as follows:

If the control input is $u <=$ gravity, the aircraft remains stationary on the ground, which can prevent the quadcopter from ''falling'' to the ground when the thrust is too small.

Otherwise, if $u >$ gravity, the four-axis acceleration is directed upwards.

Second, the PID control system adopts the positional PID control formula mentioned in the second part of this article; thus, u is given by equation (2).

After adjusting the parameters manually according to the empirical method, it can be seen that the range of k_p, k_i and k_d of the quadcopter control system is (0, 2), (0, 1), (0, 0.1)

First, the MA is used to set the parameters, and the result is as follows:

Let $MT = 10$, $N_C = 5$, $a = 0.1$, $b = 0.1$, and $(c, d) = (-1, 1)$

From Figure 11, it can be seen that due to the influence of the step size parameter setting and the set maximum number of iterations, the basic MA is not very effective when applied to the system. If the set parameters are not in place, good results cannot be obtained, showing the setting effect.

Next, the MM-MADRL algorithm proposed in this article is used to automatically adjust the three parameters of the PID controller. The specific initial data input are as follows:

''Somersault'' is set to start the interval update after 200 rounds. For example, the original value of k_p ranges from 0 to 2, and the optimal value for multiple rounds is $best\_kp\_list =$ [0.5, 0.6, 0.7, 0.8, 0.99, 1.3, 0.44, 0.56, 1.1, 1.5, 0.9]. The length of the list is 10. The last 5 numbers are taken for the interval update.

Python example code somersault step

best_kp_lb = min(best_kp_list[-5:])= 0.44
best_kp_ub = max(best_kp_list[-5:]) = 1.5

The upper bound of the k_p interval is
init_pid[0][1] = init_pid[0][1] –
(init_pid[0][1]-best_kp_Interval[1])0.1
= 2-(2-1.5)0.1
= 1.95
The lower bound of the k_p interval is
init_pid[0][0] = init_pid[0][0] –
(init_pid[0][0]-best_kp_Interval[0])0.1
= 0-(0-0.44)0.1
= 0.044

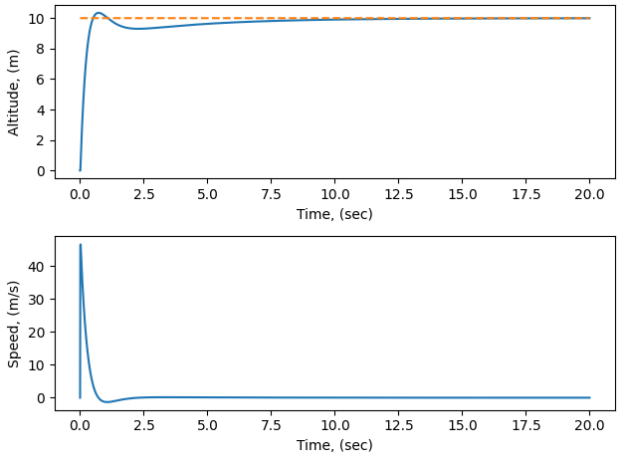


FIGURE 11. MA tuning result for a height of 10 m.

TABLE 7. MM-MADRL parameter setting.

The range of k_p is (0, 2)	The range of k_i is (0, 1)
The range of k_d is (0, 0.1)	Set the target height to 10
M=8	N_C =200
MT= 10000 (This parameter does not need to be set specifically, because the algorithm itself has a stopping condition according to the reward)	Objective function update coefficient = 0.001
Actor network learning rate=0.0001	Critic network learning rate=0.0001
Actions = 3	State = 3
shared_reward = True	t_0 =0

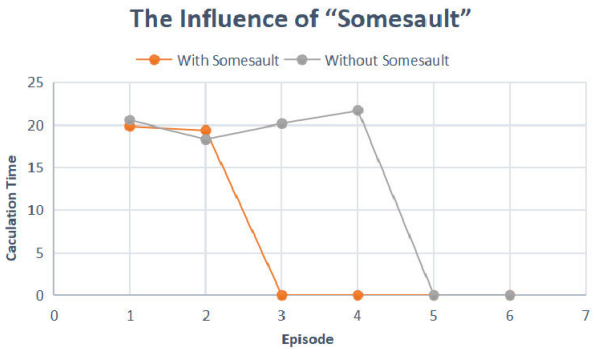


FIGURE 12. Influence of somersault.

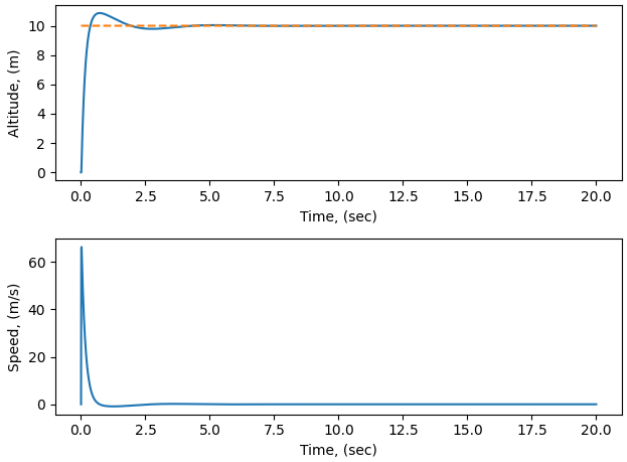


FIGURE 13. MM-MADRL tuning result for a height of 10 m.

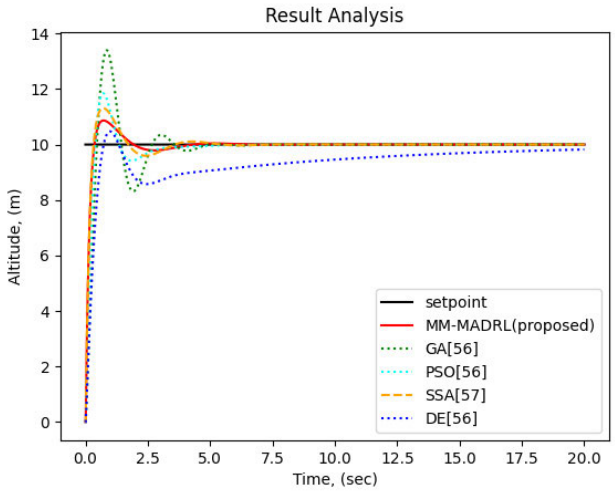


FIGURE 14. Comparison of simulation results between MM-MADRL and the GA, PSO, SSA and DE at a set height of 10 m.

Only one parameter is listed here; the other parameters use the same method. The individual monkey jumps to a smaller cube area. Finally, the interval changes from [0, 2] to [0.044, 1.95].

Figure 12 shows that ''Somersault'' can increase the speed of the algorithm. As shown in the figure, when there is ''somersault'', the calculation is completed at the third episode.

Then, the simulation result of MM-MADRL is as follows: From Figure 13, Table 8 can be obtained.

In the following, the GA, PSO, SSA and DE are used to set the parameters in the same search environment. The resulting figure is compared with that of the MM-MADRL algorithm proposed in this article, and the results obtained are shown in Figure 14.

TABLE 8. Simulation result data.

H=10 m	
Rise time t_r (sec)	0.325
Percent overshoot	4.8%
Time to steady state (sec)	4.242
k_p	1.56
k_i	1
k_d	0.02

TABLE 9. Simulation result data.

Name	t_r (sec)	Percent over shoot	Time to steady state t_s (sec)	k_p	k_i	k_d	Calculation time (sec)
MM-MADRL	0.360	8.7%	4.242	1.56	1	0.02	40.21
GA	0.460	33%	4.922	1.60	0.31	0.01	47.94
PSO	0.400	18.4%	7.043	1.84	0.56	0.01	45.28
DE	0.730	4.7%	>20	0.95	0.35	0.001	100.09
SSA	0.320	17%	4.301	1.73	1	0.03	42.07

The results obtained from Figure 14 are shown in Table 9.

In the simulation result project, three key data points, including overshoot, response time and time to steady state, are included. Overshoot is the ratio of the instantaneous maximum deviation value of the adjusted amount to the steady state value. For the system, the overshoot should be minimized because if the overshoot is too large, the aircraft system cannot meet its control requirements. The response time is an intuitive reflection of the influence of the given parameters on the control system. A shorter response time proves that the system responds faster and better. The time to reach the steady state reflects the time for the system to meet the control requirements and stabilize. A shorter time proves that the control effect of the given parameter is better, and the time for the system to reach the goal is shorter. In summary, the simulation system of this article needs to make the system react quickly after the three parameters are given and reach a stable state in a short time, and the overshoot should be as small as possible. In fact, the relationship between the three parameters of PID control and rise time, setting time and percentage of overshoot is described in detail in the literature [55].

It can be seen from Figure 14 and Table 9 that several algorithms can control the four-axis flight system, but MM-MADRL has obvious advantages in all aspects.

Next, to test the stability and accuracy of MM-MADRL, let us set the height to $H = 20$ m and $H = 40$ m and use MM-MADRL to adjust the parameters, yielding the following results, as shown in Figures 15 and 16:

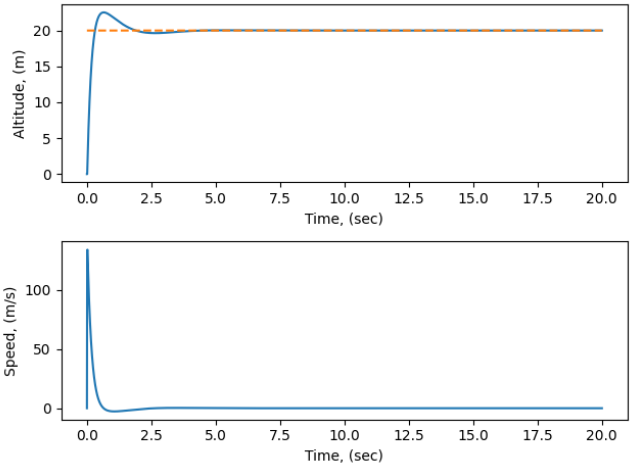


FIGURE 15. MM-MADRL tuning result for a height of 20 m.

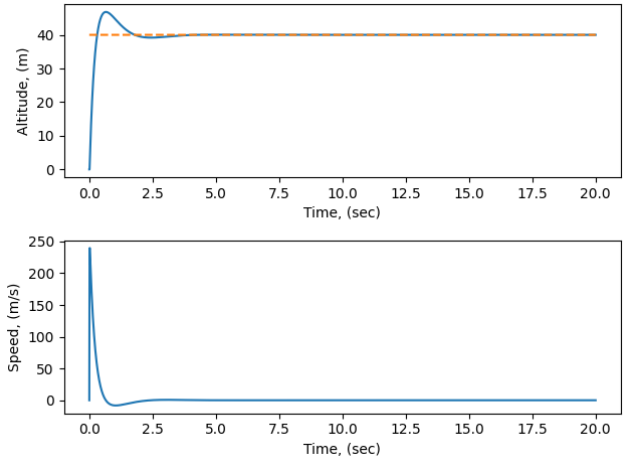


FIGURE 16. MM-MADRL tuning result for a height of 40 m.

TABLE 10. Simulation result data.

	H=20 m	H=40 m
Rise time t_r (sec)	0.320	0.300
Percent overshoot	12.6%	17%
Time to steady state (sec)	4.302	4.062
k_p	1.92	2
k_i	1	0.89
k_d	0.02	0.02

According to Figures 15 and 16, the specific data are shown in Table 10.

Table 10 shows the specific data for $H = 20$ and $H = 40$. The proposed MM-MADRL algorithm can successfully tune the PID parameters even after changing the height.

The test results show that when the MM-MADRL algorithm performs PID parameter automatic tuning, the tuning effect is good, and the stability and accuracy are excellent.

The simulation results show that the MM-MADRL algorithm proposed in this paper can perform the automatic tuning of PID parameters in a nonlinear system and achieves good results, with excellent performance in all aspects.

VI. CONCLUSION AND DISCUSSION

In this paper, under the framework of the basic monkey swarm algorithm combined with multiagent deep reinforcement learning, i.e., the proposed MM-MADRL algorithm, the simulation results show that this algorithm has a good effect in the automatic adjustment of PID parameters in a nonlinear system. Additionally, the stability and accuracy are very good. Compared with traditional intelligent algorithms, it has certain advantages. Thus, it provides a new direction for PID parameter tuning.

However, based on the simulation part, it can be seen that this research also has some areas for improvement. First, in terms of the average calculation time, although the final result obtained by the algorithm is excellent, there is still more room for improvement in the average convergence time. Second, for the automatic tuning of PID parameters, the number of parameters is small. Therefore, to improve the MM-MADRL algorithm and use it to solve other more complex and higher-dimensional systems or problems will be a future research direction.

Therefore, in future work, more in-depth improvements to the somersault step of the MM-MADRL algorithm, establishing better connections between intelligent individuals and applying the algorithm to other higher-dimensional systems will become the main directions of future research.

REFERENCES

[1] J. Pongfai, X. Su, H. Zhang, and W. Assawinchaichote, "A novel optimal PID controller autotuning design based on the SLP algorithm," *Expert Syst.*, vol. 37, pp. 1–15, Apr. 2019.

[2] M. A. Johnson and M. H. Moradi, "PID control," in *The Control Handbook*. Piscataway, NJ, USA: IEEE Press, 1996, pp. 198–209.

[3] M. Jamil, A. Waris, S. O. Gilani, B. A. Khawaja, M. N. Khan, and A. Raza, "Design of robust higher-order repetitive controller using phase lead compensator," *IEEE Access*, vol. 8, pp. 30603–30614, 2020.

[4] W. Assawinchaichote, "A non-fragile H_∞ output feedback controller for uncertain fuzzy dynamical systems with multiple time-scales," *Int. J. Comput., Commun. Control*, vol. 7, no. 1, pp. 8–19, 2012.

[5] N. Kaewpraek and W. Assawinchaichote, "H_∞ fuzzy state-feedback control plus state-derivative-feedback control synthesis for photovoltaic systems," *Asian J. Control*, vol. 18, no. 4, pp. 1441–1452, Jul. 2016.

[6] J. Pongfai, W. Assawinchaichote, P. Shi, and X. Su, "Novel D-SLP controller design for nonlinear feedback control," *IEEE Access*, vol. 8, pp. 128796–128808, 2020.

[7] A. Sungthong and W. Assawinchaichote, "Particle swam optimization based optimal PID parameters for air heater temperature control system," *Procedia Comput. Sci.*, vol. 86, pp. 108–111, Jan. 2016.

[8] S. Ruangsang and W. Assawinchaichote, "A novel robust H_∞ fuzzy state feedback plus state-derivative feedback controller design for nonlinear time-varying delay systems," *Neural Comput. Appl.*, vol. 36, pp. 6303–6318, Oct. 2019.

[9] J. Günther, E. Reichensdörfer, P. M. Pilarski, and K. Diepold, "Interpretable PID parameter tuning for control engineering using general dynamic neural networks: An extensive comparison," 2019, *arXiv:1905.13268*. [Online]. Available: http://arxiv.org/abs/1905.13268

[10] J. Fišer and P. Zítek, "PID controller tuning via dominant pole placement in comparison with ziegler-nichols tuning," *IFAC-PapersOnLine*, vol. 52, no. 18, pp. 43–48, 2019.

[11] N. P. Putra, G. J. Maulany, F. X. Manggau, and P. Betaubun, "Attitude quadrotor control system with optimization of PID parameters based on fast genetic algorithm," *Int. J. Mech. Eng. Technol.*, vol. 10, no. 1, pp. 335–343, 2019.

[12] J. Xu, "An expert PID control algorithm based on anti-integration saturation," in *Proc. IEEE 2nd Adv. Inf. Technol., Electron. Autom. Control Conf. (IAEAC)*, Mar. 2017, pp. 1536–1539.

[13] F. Kang and Y. B. Liang, "Research on modeling and simulation of expert_PID controlled servo system based on matlab/s-function," in *Applied Mechanics and Materials*. Kapellweg, Switzerland: Trans Tech, vol. 347, pp. 604–609, 2013. [Online]. Available: https://www.scientific.net/Home/Contacts

[14] B. Zhou, S. Xie, and J. Hui, "H-infinity control for T-S aero-engine wireless networked system with scheduling," *IEEE Access*, vol. 7, pp. 115662–115672, 2019.

[15] M. Farahani, S. Ganjefar, and M. Alizadeh, "Intelligent control of SSSC via an online self-tuning PID to damp the subsynchronous oscillations," in *Proc. 20th Iranian Conf. Electr. Eng. (ICEE)*, May 2012, pp. 336–341.

[16] I. Carlucho, M. De Paula, S. A. Villar, and G. G. Acosta, "Incremental Q-learning strategy for adaptive PID control of mobile robots," *Expert Syst. Appl.*, vol. 80, pp. 183–199, Sep. 2017.

[17] A. G. Alexandrov and M. V. Palenov, "Adaptive PID controllers: State of the art and development prospects," *Autom. Remote Control*, vol. 75, no. 2, pp. 188–199, Feb. 2014.

[18] Y. Liao, L. Wang, Y. Li, Y. Li, and Q. Jiang, "The intelligent control system and experiments for an unmanned wave glider," *PLoS ONE*, vol. 11, no. 12, Dec. 2016, Art. no. e0168792.

[19] Z. Jing, "Application and study of expert PID intelligent control," in *Proc. IOP Conf. Mater. Sci. Eng.*, 2019, vol. 563, no. 4, Art. no. 042084.

[20] C. Vorrawan, W. Assawinchaichote, Y. Shi, and X. Su, "Fuzzy-modeled prescribed performance integral controller design for nonlinear descriptor system with uncertainties," *IEEE Access*, vol. 8, pp. 89520–89533, 2020.

[21] S. Ruangsang and W. Assawinchaichote, "Control of nonlinear Markovian jump system with time varying delay via robust H_∞ fuzzy state feedback plus state-derivative feedback controller," *Int. J. Control, Autom. Syst.*, vol. 17, no. 9, pp. 2414–2429, 2019.

[22] F.-J. Lin, H.-J. Shieh, L.-T. Teng, and P.-H. Shieh, "Hybrid controller with recurrent neural network for magnetic levitation system," *IEEE Trans. Magn.*, vol. 41, no. 7, pp. 2260–2269, Jul. 2005.

[23] V. Kachitvichyanukul, "Comparison of three evolutionary algorithms: GA, PSO, and DE," *Ind. Eng. Manage. Syst.*, vol. 11, no. 3, pp. 215–223, Sep. 2012.

[24] E. Anene and G. K. Venayagamoorthy, "PSO tuned flatness based control of a magnetic levitation system," in *Proc. IEEE Ind. Appl. Soc. Annu. Meeting*, Oct. 2010, pp. 1–5.

[25] G. Chen, Z. Li, Z. Zhang, and S. Li, "An improved ACO algorithm optimized fuzzy PID controller for load frequency control in multi area interconnected power systems," *IEEE Access*, vol. 8, pp. 6429–6447, 2020.

[26] S. Kansit and W. Assawinchaichote, "Optimization of PID controller based on PSOGSA for an automatic voltage regulator system," *Procedia Comput. Sci.*, vol. 86, pp. 87–90, Jan. 2016.

[27] S. K. Nguang, W. Assawinchaichote, P. Shi, and Y. Shi, "H_∞ fuzzy filter design for uncertain nonlinear systems with Markovian jumps: An LMI approach," in *Proc. Amer. Control Conf.*, vol. 3, 2005, pp. 1799–1804.

[28] S. Panda, B. Mohanty, and P. K. Hota, "Hybrid BFOA–PSO algorithm for automatic generation control of linear and nonlinear interconnected power systems," *Appl. Soft Comput.*, vol. 13, no. 12, pp. 4718–4730, Dec. 2013.

[29] K. Premkumar and B. V. Manikandan, "Fuzzy PID supervised online ANFIS based speed controller for brushless DC motor," *Neurocomputing*, vol. 157, pp. 76–90, Jun. 2015.

[30] R. K. Sahu, S. Panda, and N. K. Yegireddy, "A novel hybrid DEPS optimized fuzzy PI/PID controller for load frequency control of multi-area interconnected power systems," *J. Process Control*, vol. 24, no. 10, pp. 1596–1608, Oct. 2014.

[31] K. Premkumar and B. V. Manikandan, "Bat algorithm optimized fuzzy PD based speed controller for brushless direct current motor," *Eng. Sci. Technol., Int. J.*, vol. 19, no. 2, pp. 818–840, Jun. 2016.

[32] R.-Q. Zhao and W.-S. Tang, "Monkey algorithm for global numerical optimization," *J. Uncertain Syst.*, vol. 2, no. 3, pp. 165–176, 2008.

[33] L. J. Ke and X. Q. Wang, *Reinforcement Learning*, 1st ed, Beijing, China: Tsinghua Univ. Press, 2019, pp. 1–176.

[34] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. ICML*, Jun. 2014, pp. 387–395.

[35] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. ICLR*, 2016, pp. 1–14.

[36] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," 2017, *arXiv:1706.02275*. [Online]. Available: http://arxiv.org/abs/1706.02275

[37] Y. D. Song, Y. Q. Zhang, and J. H. Zhang, "Research on the golden section method used in the optimization and tuning technology of PID parameters," *J. Xi'an Univ. Eng. Sci. Technol.*, vol. 21, no. 2, pp. 262–266, 2007.

[38] J. L. Meza, V. Santibanez, R. Soto, and M. A. Llama, "Fuzzy self-tuning PID semiglobal regulator for robot manipulators," *IEEE Trans. Ind. Electron.*, vol. 59, no. 6, pp. 2709–2717, Jun. 2012.

[39] J. Yu, P. Shi, H. Yu, B. Chen, and C. Lin, "Approximation-based discrete-time adaptive position tracking control for interior permanent magnet synchronous motors," *IEEE Trans. Cybern.*, vol. 45, no. 7, pp. 1363–1371, Jul. 2015.

[40] A. Noshadi, J. Shi, W. S. Lee, P. Shi, and A. Kalam, "Optimal PID-type fuzzy logic controller for a multi-input multi-output active magnetic bearing system," *Neural Comput. Appl.*, vol. 27, no. 7, pp. 2031–2046, Oct. 2016.

[41] B. Luo, Y. Yang, and D. Liu, "Adaptive *Q*-learning for data-based optimal output regulation with experience replay," *IEEE Trans. Cybern.*, vol. 48, no. 12, pp. 3337–3348, Apr. 2018.

[42] C.-F. Juang, "Combination of online clustering and Q-value based GA for reinforcement fuzzy system design," *IEEE Trans. Fuzzy Syst.*, vol. 13, no. 3, pp. 289–302, Jun. 2005.

[43] Q. Wei, L. F. Lewis, Q. Sun, P. Yan, and R. Song, "Discrete-time deterministic *Q*-learning: A novel convergence analysis," *IEEE Trans. Cybern.*, vol. 47, no. 5, pp. 1224–1237, May 2017.

[44] H. Mao, Z. Zhang, Z. Xiao, and Z. Gong, "Modelling the dynamic joint policy of teammates with attention multi-agent DDPG," 2018, *arXiv:1811.07029*. [Online]. Available: http://arxiv.org/abs/1811.07029

[45] E. Wei, D. Wicke, and D. Freelan, "Multiagent soft Q-learning," 2018, *arXiv:1804.09817*. [Online]. Available: https://arxiv.org/abs/1804.09817

[46] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.

[47] R. E. Wang, M. Everett, and J. P. How, "R-MADDPG for partially observable environments and limited communication," 2020, *arXiv:2002.06684*. [Online]. Available: http://arxiv.org/abs/2002.06684

[48] J. Han, C.-H. Wang, and G.-X. Yi, "Cooperative control of UAV based on multi-agent system," in *Proc. IEEE 8th Conf. Ind. Electron. Appl. (ICIEA)*, Jun. 2013, pp. 96–101.

[49] K. Shao, Y. Zhu, and D. Zhao, "StarCraft micromanagement with reinforcement learning and curriculum transfer learning," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 3, no. 1, pp. 73–84, Feb. 2019.

[50] J. E. Summers, J. M. Trader, C. F. Gaumond, and J. L. Chen, "Deep reinforcement learning for cognitive sonar," *J. Acoust. Soc. Amer.*, vol. 143, no. 3, p. 1716, Apr. 2018.

[51] X. L. Wei, X. L. Huang, T. Lu, and G. G. Song, "An improved method based on deep reinforcement learning for target searching," in *Proc. 4th Int. Conf. Robot. Autom. Eng. (ICRAE)*, Nov. 2019, pp. 130–134.

[52] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," *arXiv:1312.5602*, Available: https://arxiv.org/abs/1312.5602

[53] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, Sep. 2015. [Online]. Available: https://arxiv.org/abs/1509.02971

[54] S. Omidshafiei, J. Pazis, C. Amato, J. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proc. Int. Conf. Mach. Learn.*, Mar. 2017, pp. 2681–2690.

[55] E. Hub. *PID Controller-Working and Tuning Methods*. Accessed: Dec.10, 2015. [Online]. Available: https://www.electronicshub.org/pid-controller-working-and-tuning-methods/

[56] Z. Bao, J. Yu, and S. Yang, *Intelligent Optimization Algorithm and its MATLAB Example*, 2nd ed. Beijing, China, Electronic Industry Press, 2018.

[57] J. Xue and B. Shen, "A novel swarm intelligence optimization approach: Sparrow search algorithm," *Syst. Sci. Control Eng.*, vol. 8, no. 1, pp. 22–34, Jan. 2020.

[58] M. Eatemadi, "Mathematical dynamics, kinematics modeling and PID equation controller of quadCopter," *Int. J. Appl. Oper. Res.*, vol. 7, no. 1, pp. 77–85, 2017.

[59] Y. Mousavi, A. Alfi, and I. B. Kucukdemiral, "Enhanced fractional chaotic whale optimization algorithm for parameter identification of isolated wind-diesel power systems," *IEEE Access*, vol. 8, pp. 140862–140875, 2020.

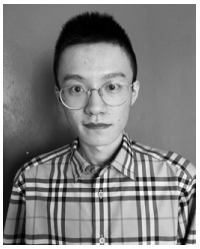
[60] J. A. T. Machado, S. M. A. Pahnehkolaei, and A. Alfi, "Complex-order particle swarm optimization," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 92, Jan. 2020, Art. no. 105448.

[61] H. Shokri-Ghaleh, A. Alfi, S. Ebadollahi, A. M. Shahri, and S. Ranjbaran, "Unequal limit cuckoo optimization algorithm applied for optimal design of nonlinear field calibration problem of a triaxial accelerometer," *Measurement*, vol. 164, Nov. 2020, Art. no. 107963.

[62] A. Alfi and M.-M. Fateh, "Intelligent identification and control using improved fuzzy particle swarm optimization," *Expert Syst. Appl.*, vol. 38, no. 10, pp. 12312–12317, Sep. 2011.

[63] A. Alfi and H. Modares, "System identification and control using adaptive particle swarm optimization," *Appl. Math. Model.*, vol. 35, no. 3, pp. 1210–1221, Mar. 2011.

[64] E. S. A. Shahri, A. Alfi, and J. A. T. Machado, "Fractional fixed-structure H_∞ controller design using augmented lagrangian particle swarm optimization with fractional order velocity," *Appl. Soft Comput.*, vol. 77, pp. 688–695, Apr. 2019.



HONGMING ZHANG was born in Kunming, Yunnan, China, in 1993. He received the B.S. degree in information and communication engineering from the University of Mea Fah Luang, Chiang Rai, Thailand, in 2018, and the M.S. degree in electronic and telecommunication engineering from the King Mongkut's University of Technology Thonburi, Bangkok, Thailand, in 2020, where he is currently pursuing the Ph.D. degree. From 2018 onwards, he researches on PID control systems, mainly in the aspects of intelligent algorithms and neural networks, to optimize the PID systems.

WUDHICHAI ASSAWINCHAICHOTE received the B.S. degree (Hons.) in electrical engineering from Assumption University, Bangkok, Thailand, in 1994, the M.E. degree in electrical engineering from Pennsylvania State University (Main Campus), PA, USA, in 1997, and the Ph.D. degree in electrical engineering from The University of Auckland, New Zealand, in 2004. He is currently an Associate Professor with the Department of Electronic and Telecommunication Engineering, King Mongkut's University of Technology Thonburi (KMUTT), Bangkok. He has published a research monograph and more than 20 research articles in international refereed journals indexed by SCI/SCIE (Clarivate Analytics). His current research interests include fuzzy control, robust control, optimal control, system and control theory, computational intelligence, and PID controller design. He also serves as an Associate Editor for the *International Journal of Innovative Computing, Information and Control*, and a Reviewer for IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, IEEE TRANSACTIONS ON FUZZY SYSTEMS, IEEE TRANSACTIONS ON CYBERNETICS, *Neural Computing and Applications*, and IEEE ACCESS.

YAN SHI received the Ph.D. degree in information and computer sciences from Osaka Electro-Communication University, Neyagawa, Japan, in 1997. He is currently a full-time Professor with the Graduate School of Science and Technology, Tokai University, Kumamoto, Japan. His research interests include fuzzy reasoning and data mining.