Zeng *et al*. *Intell. Robot*. **2025,** *5*(2), 276-91
DOI: 10.20517/ir.2025.14

Intelligence & Robotics

Research Article

Open Access

An effective fault-tolerant control with slime mold algorithm for unmanned underwater vehicle

Tianhong Zeng[1], Daqi Zhu[1], Chunhua Gu[1], Simon X. Yang[2]

[1]Research Institute of Underwater Vehicles and Intelligent Systems, University of Shanghai for Science and Technology, Shanghai, 200093, China.
[2]The Advanced Robotics and Intelligent Systems Laboratory, University of Guelph, Guelph, ON N1G2W1, Canada.

Correspondence to: Prof. Daqi Zhu, Research Institute of Underwater Vehicles and Intelligent Systems, University of Shanghai for Science and Technology, Jungong Road 516, Shanghai, 200093, China. E-mail: zdq367@aliyun.com; ORCID: 0000-0001-7252-4952

How to cite this article: Zeng, T.; Zhu, D.; Gu, C.; Yang, S. X. An effective fault-tolerant control with slime mold algorithm for unmanned underwater vehicle. *Intell. Robot.* **2025**, *5*(2), 276-91. http://dx.doi.org/10.20517/ir.2025.14

Received: 3 Dec 2024 **First Decision:** 13 Jan 2025 **Revised:** 22 Jan 2025 **Accepted:** 26 Feb 2025 **Published:** 27 Mar 2025

Academic Editor: Jinhai Li, Yueying Wang **Copy Editor:** Pei-Yun Wang **Production Editor:** Pei-Yun Wang

Abstract

The proposed fault-tolerant control strategy based on the slime mold algorithm (FTC-SMA) enhances the resilience of multi-thruster unmanned underwater vehicles against thrust loss. In the event of a propulsion system failure, the strategy enables rapid thrust redistribution to restore the original torque and sailing direction, even in the event of a catastrophic thruster failure. This strategy follows the physical limits of the thruster and can effectively solve the over-actuated problem. The effectiveness, efficiency, and stability of FTC-SMA are confirmed through simulation experiments under various fault conditions, demonstrating significant improvements over other algorithms such as particle swarm optimization and grasshopper optimization algorithm.

Keywords: Multi-thruster, fault-tolerant control, slime mold algorithm (SMA), unmanned underwater vehicle (UUV)

1. INTRODUCTION

With the growth of control applications in complex environments, fault-tolerant control (FTC) has found widespread applications across various fields. During the past three decades, extensive and comprehensive studies have been conducted on FTC, particularly in the aviation, spacecraft, automobiles, and industrial manufacturing sectors[1–4]. These investigations focus on developing systems capable of accommodating component failures during operation, ensuring overall device stability with minimal or acceptable declines in performance. However, research on FTC in underwater vehicles remains limited, largely due to the complexities associated with the underwater environment and marine vehicle systems[5]. Underwater vehicles play a crucial role in a variety of underwater tasks, including deep-sea exploration, ocean rescue, resource exploitation, and underwater target search. The propulsion system is essential for the efficient operation of these vehicles, where the thrusters that consist of the propulsion system control the underwater vehicle's motion by adjusting the torque across six degrees of freedom (DOFs)[6,7]. Thrusters exposed to the elements for a long time are easily affected by the complex marine environment, causing many faults such as mud and aquatic plant attachment, circuit short circuits, thruster collision damage, *etc.*, resulting in thrust loss of varying degrees[8]. Therefore, FTC of underwater propulsion systems is crucial for ensuring robust and efficient navigation of the vehicle[9,10].

Numerous scholars dedicate their efforts to maintaining system stability and preventing losses under various potential failure conditions[11−14]. However, the inherent diversity and uncertainty of failures often pose challenges in achieving comprehensive consideration. Some research suggests an alternative approach by utilizing an excess number of thrusters to optimally redistribute the control system, enabling mission completion and sustaining high performance despite vehicle malfunctions[15,16]. For instance, in the event of a thruster failure during an underwater mission, installing a propulsion system equal to or greater than the DOF provides additional flexibility to compensate for the torque required across all six DOFs. To facilitate thrust reallocation, the weighted pseudo-inverse algorithm is commonly employed for calculations[17]. However, since the physical limitations of the thrusters are not involved, the results obtained by the algorithm may exceed the feasible range, resulting in an overdrive problem; that is, the control input exceeds the actual maximum control range.

To address the challenge of vehicle over-actuation, researchers[18] proposed a fault diagnosis method combining self-organizing mapping and fuzzy logic clustering. They utilized the weighted pseudo-inverse to determine the reassigned control vector, employing truncation or scaling as approximation methods to ensure solution feasibility. However, these two approximation methods roughly force the output to be limited within the feasible range after the calculation is completed. When the calculation result is far beyond the feasible range, direct truncation or scaling will not guarantee accuracy.

Combining optimization methods to find the optimal solution within the allowed range is the current mainstream research direction. Researchers have achieved good results in the FTC field using genetic algorithms, neural network methods, greedy algorithms, *etc.* However, high-quality data input and long computing time are not easy to apply in underwater submersibles[3,19]. Meta-heuristic algorithms have gained prominence in various applied disciplines[20−23]which can achieve higher performance in a shorter time. Moreover, the solution space for thruster torque redistribution is often uncertain or infinite[24]. In such cases, finding the optimal solution by traversing the entire solution space may be impractical. Meta-heuristic algorithms address this by detecting an approximate optimal solution through random sampling within a vast solution space. Zhu *et al.* proposed a particle swarm optimization (PSO)-based thruster reconstruction control method to find solutions within the feasible space without relying on truncation or scaling approximation methods to limit the control vector[25]. This method demonstrated a small control error compared to the weighted pseudo-inverse method. Tian *et al.*, considering the uncertainty of ocean currents, introduced an improved cooperative PSO (CPSO) algorithm for FTC[26]. However, due to its simple structure, PSO often struggles to meet the accuracy requirements for complex tasks[27]. Zhu *et al.* proposed grasshopper optimization algorithm (GOA), which obtained satisfactory results[28,29]. However, there is still room for improvement in real-time processing and accuracy.

This paper proposes for the first time a practical case application of combining the slime mold algorithm (SMA) with FTC to solve the thrust loss fault of underwater vehicle propellers. By quantifying the thrust loss of each

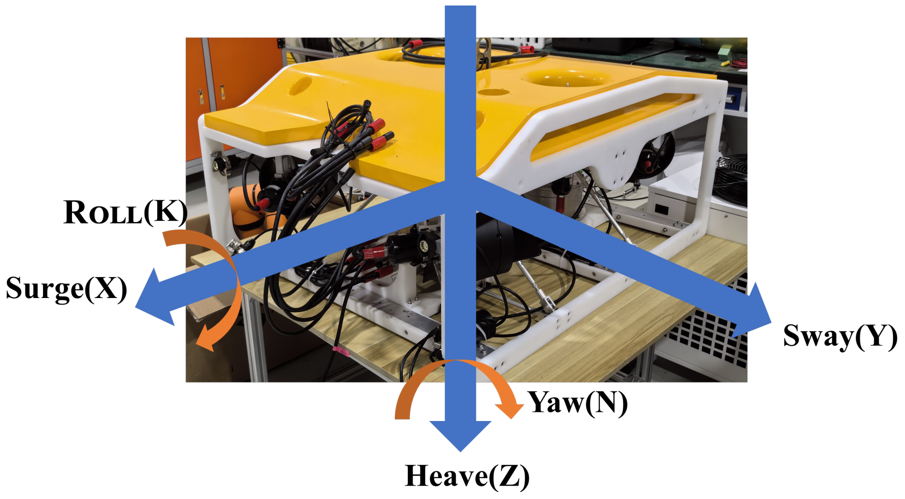

Figure 1. "Haijian" UUV. UUV: Unmanned underwater vehicle.

propeller, SMA is used to quickly and effectively redistribute the thrust. Compared with traditional methods such as T-approximation or S-approximation, the SMA algorithm can solve the drive saturation problem, and compared with the most advanced swarm intelligence optimization method GOA currently used in underwater submersibles, it achieves higher accuracy and faster computational efficiency.

The contributions of this paper are as follows:
The fault-tolerant control strategy based on the slime mold algorithm (FTC-SMA) effectively solves over-actuation, a common challenge in conventional control systems.
The proposed FTC-SMA not only boasts fast convergence speeds but also maintains high accuracy. This dual advantage allows for the quick settling of redistributing thrust to the thrusters.
Three fault cases, single-fault case, double-fault case and quadruple-fault case, were designed and simulated to demonstrate the adaptability and reliability of FTC-SMA under different fault conditions.

The article is organized as follows. Section 2 provides an in-depth introduction to the thruster configuration in unmanned underwater vehicles (UUVs) and the underlying principles of FTC for underwater vehicles. Section 3 explores the characteristics of the SMA and details its integration with FTC methodologies. Section 4 presents and discusses the outcomes of computer simulations applying the proposed algorithm to various fault conditions, and compares results obtained under different fault scenarios using computer simulation algorithms. Section 5 summarizes key findings from the simulations, draws conclusions based on the results and discusses potential implications for the field of FTC in UUVs.

2. THRUSTER CONFIGURATION AND PROBLEM STATEMENT

In this section, a detailed model and illustration of the propulsion system of the UUV being developed in our lab is provided. The operational context of FTC in the vehicle is elucidated and analyzed comprehensively.

2.1. Thruster configuration

"Haijian" submarine is a small search and rescue UUV that can perform exploration, detection and grabbing operations [Figure 1] [30]. The vehicle features a configuration with six thrusters distributed around its structure, as depicted in Figure 2. Each thruster is denoted by Ti, where i ranges from 1 to 6. Specifically, $T1$, $T2$, $T3$, $T4$ are four transverse thrusters positioned horizontally at a 30° angle with the X-axis, all situated in the XoY plane. $T5$, $T6$ are two vertical thrusters positioned longitudinally, both located in the YoZ plane. Each individual

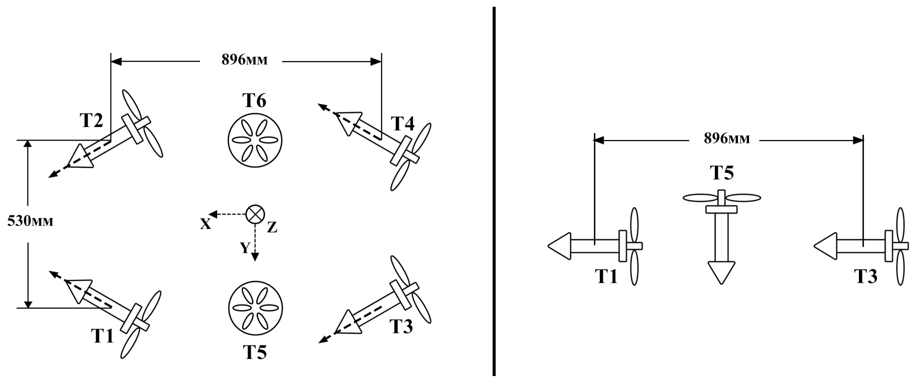

Figure 2. Top view and side view of UUV thruster system distribution. UUV: Unmanned underwater vehicle.

thruster is responsible for generating the necessary torque across five DOFs (lack of the pitch DOF), resulting in a comprehensive set of torques. These torques, corresponding to different DOFs, collectively contribute to achieving the desired motion and attitude of the UUV. Throughout this article, the torque vector τ serves as a representative notation for these five torques,

$$\tau = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ \tau_k \\ \tau_n \end{bmatrix} \tag{1}$$

where x, y, z, k, n represent axes constructed with five DOFs (surge, sway, heave, roll, yaw), respectively.

Combined with the position structure information of the thruster, the relationship between its torque vector and thruster force is as follows:

$$\tau = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ \tau_k \\ \tau_n \end{bmatrix} = \begin{bmatrix} 0.866T_1 + 0.866T_2 + 0.866T_3 + 0.866T_4 \\ -0.5T_2 - 0.5T_3 + 0.5T_1 + 0.5T_4 \\ T_5 + T_6 \\ 0.265T_5 - 0.265T_6 \\ 0.453T_1 + 0.453T_3 - 0.453T_2 - 0.453T_4 \end{bmatrix} \tag{2}$$

where $T_1, T_2, T_3, T_4, T_5, T_6$ are the forces applied to the six thrusters around the vehicle. The transformation of the torque vector and the force vector can be written as:

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ \tau_k \\ \tau_n \end{bmatrix} = \begin{bmatrix} 0.866 & 0.866 & 0.866 & 0.866 & 0 & 0 \\ 0.5 & -0.5 & -0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0.265 & -0.265 \\ 0.453 & -0.453 & 0.453 & -0.453 & 0 & 0 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{bmatrix} = BT \tag{3}$$

where B is the thruster control matrix and T is the thruster force matrix.

The whole propulsion system is composed of two different types of thrusters, among which $T1, T2, T3, T4$ are the same type, and $T5$,$T6$ are the same type. It is assumed that they have the maximum thrust T_{m1} and T_{m2}, respectively, and thus the maximum torque vector τ_m is derived

$$\tau_m = \begin{bmatrix} \tau_{\mathrm{xm}} \\ \tau_{ym} \\ \tau_{zm} \\ \tau_{km} \\ \tau_{nm} \end{bmatrix} = \begin{bmatrix} 4 \times 0.866T_{m1} \\ 4 \times 0.5T_{m1} \\ 2 \times T_{m2} \\ 2 \times 0.265T_{m2} \\ 4 \times 0.453T_{m1} \end{bmatrix} \tag{4}$$

Dividing Equation (2) by Equation (4) limits the output in a certain range of -1 to 1, and set $\overline{\tau} = \tau/\tau_m, \overline{T} = T/T_m$; the vector is transformed into

$$\begin{aligned} \overline{\tau} = \begin{bmatrix} \overline{\tau_{\mathrm{x}}} \\ \overline{\tau_y} \\ \overline{\tau_z} \\ \overline{\tau_k} \\ \overline{\tau_n} \end{bmatrix} = \begin{bmatrix} \tau_{\mathrm{x}}/\tau_{\mathrm{xm}} \\ \tau_{\mathrm{y}}/\tau_{\mathrm{ym}} \\ \tau_{\mathrm{z}}/\tau_{\mathrm{zm}} \\ \tau_k/\tau_{\mathrm{km}} \\ \tau_n/\tau_{\mathrm{nm}} \end{bmatrix} &= \begin{bmatrix} (0.866T_1 + 0.866T_2 + 0.866T_3 + 0.866T_4)/4 \times 0.866T_{m1} \\ (-0.5T_2 - 0.5T_3 + 0.5T_1 + 0.5T_4)/4 \times 0.5T_{m1} \\ (T_5 + T_6)/2 \times T_{m2} \\ (0.265T_5 - 0.265T_6)/2 \times 0.265T_{m2} \\ (0.453T_1 + 0.453T_3 - 0.453T_2 - 0.453T_4)/4 \times 0.453T_{m1} \end{bmatrix} \\ &= \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 & 0 & 0 \\ 0.25 & -0.25 & -0.25 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0.5 & -0.5 \\ 0.25 & -0.25 & 0.25 & -0.25 & 0 & 0 \end{bmatrix} \begin{bmatrix} T_1/T_{m1} \\ T_2/T_{m1} \\ T_3/T_{m1} \\ T_4/T_{m1} \\ T_5/T_{m2} \\ T_6/T_{m2} \end{bmatrix} = \overline{B} \begin{bmatrix} \overline{T_1} \\ \overline{T_2} \\ \overline{T_3} \\ \overline{T_4} \\ \overline{T_5} \\ \overline{T_6} \end{bmatrix} = \overline{BT} \end{aligned} \tag{5}$$

Finally, the conversion between thruster forces and torques was achieved and standardized, limiting the range of all torques and forces in the equation to -1 to 1; this enables better understanding and easier visualization of the problem.

All physical parameters are removed from the matrix during the normalization process. The compact form of $\overline{B}$ simplifies calculations and, as will be shown later, leads to the very simple representation of the SMA and a clear geometric interpretation of the control allocation problem.

In order to quantify the damage of each thruster, the thruster weight matrix is usually defined as

$$W = \begin{bmatrix} w_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & w_6 \end{bmatrix} \tag{6}$$

where w_i is the corresponding weight value of the thruster; if any thruster experiences a power loss, its corresponding weight value will be reduced according to the degree of power loss. For instance, $w_i = 1$ when all thrusters are operating properly, and $w_2 = 0.7$ when $T2$ thruster fails and can only operate at 70% of maximum force or torque.

Combine Equation (5) with Equation (6) to get the following transition

$$\overline{\tau} = \overline{B}W\overline{T} = \overline{B}T^* \tag{7}$$

where $\overline{T}$ is the thruster control parameters, derived and solved through optimization; T^* are the actual thruster control parameters sent to the system.

2.2. Problem statement

This section delineates the FTC problem within the UUV multi-thruster system, elucidating the target requirements and constraint information pivotal to the control process.

For the UUV multi-thruster system, the ideal FTC is through

$$\|\mathrm{e}\| = \|\overline{\tau_d} - \overline{\tau}\| \to 0 \tag{8}$$

$$\|\theta_\mathrm{e}\| = \arccos \frac{\overline{\tau_d} \cdot \overline{\tau}}{\|\overline{\tau_d}\| \cdot \|\overline{\tau}\|} \to 0 \tag{9}$$

where $\overline{\tau_d}$ represents the expected state, $\overline{\tau}$ represents the actual obtained state, the magnitude error $\|\mathrm{e}\|$ signifies the modulus of the difference vector between the expected state and the actual state, and the direction error $\|\theta_\mathrm{e}\|$ indicates the angle between the expected state and the actual state. The two performance indicators specifically measure the accuracy of the thrust control of the UUV and the accuracy of its directional control. When designing the fault-tolerant control, these two performance indicators should be as small as possible.

In practical applications, due to limitations in thruster capabilities, such as the inability to provide infinite drive input for completing voyages, an over-actuated phenomenon occurs. Therefore, in optimization calculations, the actual torque or force that the thruster can output must be physically constrained to ensure reliable navigation. The maximum thrust of the thruster is determined by the maximum torque provided by the vehicle, which serves as a fundamental constraint in the vehicle control problem.

In subsequent experiments, both the maximum torque and the maximum thrust were constrained within the range of -1 to 1. These variables were employed to quantify their influence on the results during the control process.

3. FTC BASED ON SMA DESIGN

This section reviews the basic principles of SMA and then explores its application in FTC.

3.1. SMA

The SMA is a novel meta-heuristic algorithm, which primarily emulates the behavior and morphological transformations of slime molds during foraging in nature[31]. The adoption of distributed computing principles in SMA allows multiple agents to simultaneously explore the solution space, enhancing search efficiency through collaborative information exchange. This distributed approach empowers SMA to facilitate the discovery of global optimal solutions while avoiding the pitfalls of local minima. The self-organizing nature of SMA, characterized by entities interacting and adapting through localized planning, contributes to its adaptability and robustness in complex and dynamic environments. Specifically, SMA comprises three key stages: approach food stage, wrap food stage, and oscillation stage.

3.1.1 Approach food

In the approach food stage, the slime mold moves towards the food source guided by the scent in the air. Its approach behavior can be mathematically given by

$$X(t+1) \equiv \begin{cases} X_b(t) + vb(W_s \times X_A(t) - X_B(t)), & r < \mathrm{p} \quad (10a) \\ vc \times X(t), & r \geq \mathrm{p} \quad (10b) \end{cases}$$

where $X(t+1)$ and $X(t)$ represent the positions of slime molds at iterations $t+1$ and t, respectively, and $X_b(t)$ denotes the positions with the highest food concentration at iteration t. $X_A(t)$ and $X_B(t)$ represent two randomly selected slime molds at iteration t. The range of vb is $[-a, a]$, where $a = \mathrm{arctanh}(1 - t/t_{\max})$, t is the current iteration number, and t_{max} is the maximum iteration number. The range of vc linearly decreases from 1 to 0. r is a random number between 0 and 1. $p = \tanh(|S(i) - DF|)$, where $i = 1, 2, ..., N$, $S(i)$ represents the fitness value of the i_{th} slime mold individual, DF represents the optimal fitness value across all iterations, and N represents the population size of slime molds. W_s represents the weight of slime molds, and it is calculated by

$$W_s(SortIndex(i)) \equiv \begin{cases} 1 + r \times \log(\dfrac{bF - S(i)}{bF - wF} + 1), & \text{condition} \quad (11a) \\ 1 - r \times \log(\dfrac{bF - S(i)}{bF - wF} + 1), & \text{others} \quad (11b) \end{cases}$$

where $condition$ represents the individuals whose fitness values rank in the top half of the group, bF, wF represent the best and worst fitness values in the current iteration, respectively, and $SortIndex$ represents the sorted fitness value sequence. During the approach food stage, the position of searching individual is updated based on variations in the optimal position Xb, fine-tuning of vb, vc and W_s. The function of r is to generate a search vector at any angle, enabling exploration of the solution space in any direction, thus increasing the likelihood of discovering an optimal solution.

3.1.2 Wrap food

The wrap food stage simulates the contraction pattern of myxomycetes vein tissue. As the concentration of food increases in venous contact, the biological oscillator produces a stronger propagation wave, accelerating the cytoplasmic flow. Equation (11) models the positive and negative feedback process between slime mold weight and food concentration. The logarithm function $log()$ is utilized to moderate the rate of change and stabilize the contraction frequency. $conditon$ simulates the process of slime mold adjusting its position according to food concentration. When the food concentration is higher, the weight of slime mold in the vicinity increases. Conversely, if the food concentration is low, slime molds will redirect their search to other areas, leading to a reduction in the weight of slime molds in that region. Based on these principles, the mathematical formula for updating the location of slime molds in this stage is:

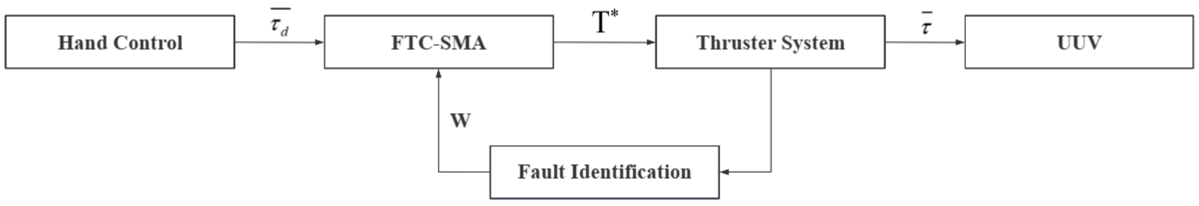

Figure 3. The structure diagram of tolerant control system.

$$X(t+1) \equiv \begin{cases} rand \times (ub - lb) + lb, & rand < z \quad (12a) \\ X_b(t) + vb(W_s \times X_A(t) - X_B(t)), & r < \mathrm{p} \quad (12b) \\ vc \times X(t), & r \geq \mathrm{p} \quad (12c) \end{cases}$$

where $rand$ and r represent random values generated within the range of 0 to 1, and ub and lb denote the upper and lower bounds of the search space, respectively. z represents the proportion of randomly distributed slime mold individuals in the total slime mold population. This parameter is utilized by the algorithm to transition between the global search stage and the local search stage.

3.1.3 Oscillation

In the grabbing food stage, slime molds utilize propagation waves generated by biological oscillators to alter the velocity of cytoplasm flow within veins. Parameters vb, vc and W_s simulate the adjustment of vein width and the oscillation frequency of biological oscillators. Consequently, slime molds approach food more gradually when the food concentration is low, and conversely, they move toward food more rapidly when encountering high-quality food. Essentially, this stage involves the update of parameters vb, vc and W_s.

vb and vc are crucial parameters in simulating oscillation, which simulate the selective behavior of slime mold. Slime molds with lower fitness engage in global search, while slime molds with higher fitness engage in local search, and the oscillation of vb and vc diffuses the search direction of slime molds. Additionally, when $rand$ is less than z, slime molds undergo random initialization. This indicates that while the SMA searches for the optimal solution, it simultaneously dispatches a smaller segment of the slime mold to explore for potentially better solutions. This strategy effectively addresses the issue of the GOA frequently becoming trapped in local optima.

3.2. FTC-SMA

In this paper, the SMA is employed to determine the optimal control vector. The tolerant control system structure is illustrated in Figure 3. The input to the system is the desired torque vector. The fault identification unit monitors the status of the thrusters, and the output is the thruster weight matrix Equation (6) that encapsulates the fault information, and the FTC-SMA makes corresponding adjustments within the limit parameters, the recalculated thrust required by each thruster is output to the thruster system to achieve the purpose of FTC. It is worth noting that, given that the main focus of this paper is FTC, we assume that the fault identification unit has accurately identified and calculated the thrust loss of each thruster when a fault occurs.

The pseudo-code of FTC-SMA is presented, which helps to understand its principle and structure better. The specific steps are as follows:

1. Given thruster weight matrix W and desired torque matrix $\overline{\tau_d}$, set parameters, and initialize $popsize$, $max\_iteration$ and the position of each slime mold. Using ten sets of six random numbers between -1

Algorithm 1 Multi-thruster FTC based on SMA algorithm

```
Input: thruster weight matrix $W$ and desired torque vector $\overline{\tau_d}$
    Initialize the parameter $popsize$, $max\_iteration$ and position of slime mold
    while $t < max\_iteration$ do
        Calculate the $||e||$, $||\theta_e||$ of each search agent by Equations (8) and (9)
        Update $||e||$, $||\theta_e||$ and $T^*$
        Calculate the $W_s$ by Equation (11) and a by $a = arctanh(1 - t/t_{max})$
        for each search portion do
            Update $p$, $vb$, $vc$
            Update positions by Equation (12)
        end for
        $t = t + 1$
    end while
Output: $||e||$, $||\theta_e||$, $T^*$
```

and 1 to represent normalized thrusters, each set is treated as a search agent.

2. Calculate fitness $||e||$, $||\theta_e||$, slime mold weight W_s, and parameter a for each search agent with Equations (8), (9) and (11), and $a = arctanh(1 - t/t_{max})$.
3. Generate random number r and evaluate the magnitude of r and parameter z: If $r < z$, update position with Equation (12a); otherwise, update parameters p, vb and vc. Evaluate the magnitude of r and parameter p. If $r < p$, update position with Equation (12b); otherwise, update position with Equation (12c).
4. Calculate fitness value after updating the position and update the global optimal solution.
5. Check termination condition; if met, output global optimal solution (force of each thruster after redistribution T^*) and fitness value $||e||$, $||\theta_e||$; otherwise, repeat steps 2-5.

4. SIMULATION AND ANALYSIS

In this section, the computer simulation results of the T-approximation, S-approximation, PSO, GOA and SMA for multi-thruster FTC are presented. The purpose is to compare their performance under identical working conditions. Various fault cases have been selected to highlight the superiority of the FTC-SMA in terms of accuracy and stability. All simulations are conducted using Python 3.9, ensuring that each run achieves optimal convergence for all algorithms. In this paper, it is assumed that there are no other faults except the thruster, such as incorrect sensor readings or controller failures. It is also assumed that the thrust loss of the thruster has been accurately identified by the Fault Identification unit and the thruster weight matrix W is output. The following subsections detail the results and errors observed in each algorithm, providing a comprehensive understanding of their strengths and weaknesses in handling multi-thruster FTC.

4.1. Single-fault case

The desired torque vector $\overline{\tau_d}$ is assigned as $[0.6,\ 0.2,\ 0.2,\ 0.0,\ 0.1]$. At the 60-second mark, thruster $T3$ experienced a failure, resulting in a 20% reduction in its power output, the corresponding $W = [1.0,\ 1.0,\ 0.8,\ 1.0,\ 1.0,\ 1.0]$. In response to this fault occurrence, various optimization algorithms, namely T-approximation, S-approximation, PSO, GOA and SMA, were employed to derive optimal power allocation solutions. The obtained calculation results are shown in Table 1.

The pseudo-inverse solution $\overline{T}$ is $[0.4526,\ 0.6526,\ 0.9474,\ 0.3474,\ 0.2,\ 0.2]$ and one component is unfeasible because $\overline{T}_3 > 0.8$. Utilizing the T-approximation method, values exceeding control constraints are truncated. Specifically, 0.9474 will be set to 0.8, so that $\overline{T}$ is adjusted to $T^* = [0.4526,\ 0.6526,\ 0.8,\ 0.3474,\ 0.2,\ 0.2]$, and the corresponding $\overline{\tau} = [0.5232,\ 0.1232,\ 0.2,\ 0.0,\ 0.0232]$. S-approximation will be obtained by $min\ \left(1/max_{i\in\{1,2,3,4,5,6\}}\ \left|\bar{T}_i/W_i\right|\right)$

Table 1. Results of single-fault case

	$\mathbf{T}^*$	$\overline{\tau}$	$\|\mathbf{e}\|$	$\|\theta_{\mathbf{e}}\|$
T-approximation	$[0.4526, 0.6526, 0.9474, 0.3474, 0.2000, 0.2000]$	$[0.5232, 0.1232, 0.2000, 0.0000, 0.0232]$	0.1331	0.1472
S-approximation	$[0.3822, 0.5511, 0.8000, 0.2933, 0.1689, 0.1689]$	$[0.4667, 0.1289, 0.1689, 0.0000, 0.0444]$	0.1640	0.0852
PSO	$[0.5551, 0.6828, 0.3381, -0.0447, 0.1276, 0.1276]$	$[0.3828, 0.1276, 0.1276, 0.0000, 0.0638]$	0.2428	0.0001
GOA	$[0.1980, 0.3408, 0.8000, 0.3724, 0.1425, 0.1426]$	$[0.4278, 0.1426, 0.1425, 0.0000, 0.0712]$	0.1926	0.0002
SMA	$[\mathbf{0.8013}, \mathbf{1.0000}, \mathbf{0.5965}, \mathbf{0.0001}, \mathbf{0.2023}, \mathbf{0.1986}]$	$[\mathbf{0.5995}, \mathbf{0.1988}, \mathbf{0.2004}, \mathbf{0.0019}, \mathbf{0.0994}]$	**0.0024**	**0.0033**

PSO: Particle swarm optimization; GOA: grasshopper optimization algorithm; SMA: slime mold algo-rithm.

Since the first component exceeds the control constraint by a greater extent, all the components times $0.8/0.9474$, so that $\overline{T}$ is adjusted to $T^* = [0.3822,\ 0.5511,\ 0.8,\ 0.2933,\ 0.1689,\ 0.1689]$, and the corresponding $\overline{\tau}$=[0.4667, 0.1289, 0.1689, 0.0, 0.0444].

Figure 4 clearly shows the results of all algorithms. Without a doubt, the SMA stands out with the best result (orange line), which closely matches the actual torque required. Errors in the T-approximation (blue line) and S-approximation (green line) are unavoidable due to the method of directly truncating and scaling the portion beyond the limit. Since only thrusters $T5$ and $T6$ affect the heave and roll axes, but they have no thrust loss, the T approximation effectively mitigates the errors of the heave and roll axes. However, in the case of the S-approximation, significant deviations from the required torque occur due to the adjustments made to all components. The GOA (black line) and PSO (magenta line) achieve similar results. Both correct direction error well, but their performance in correcting magnitude errors is less effective. This implies that the adjusted torque output could significantly decrease the robot's operational efficiency. However, after optimization using the SMA method, the torque output errors for each axis are minimal and can essentially be disregarded. The force distribution among the thrusters has been successfully reallocated within acceptable limits.

4.2. Double-fault case

The desired torque vector $\overline{\tau_d}$ is also assigned as $[0.6,\ 0.2,\ 0.2,\ 0.0,\ 0.1]$. At the 60-second mark, thrusters $T2$ and $T3$ experienced a failure, resulting in a 30% and 20% reduction in its power output, respectively; the corresponding $W = [1.0,\ 0.7,\ 0.8,\ 1.0,\ 1.0,\ 1.0]$. The obtained calculation results are shown in Table 2.

The pseudo-inverse solution $\overline{T}$ is $[0.5086,\ 0.7086,\ 0.8914,\ 0.2914,\ 0.2,\ 0.2]$ and two components are unfeasible because $\overline{T}_2 > 0.7, \overline{T}_3 > 0.8$. Utilizing the T-approximation method, values exceeding control constraints are truncated. Specifically, 0.7086 will be set to 0.7, and 0.8914 will be set to 0.8, so that $\overline{T}$ is adjusted to $T^* = [0.5086,\ 0.7,\ 0.8,\ 0.2914,\ 0.2,\ 0.2]$, and the corresponding $\overline{\tau} = [0.4825,\ 0.0825,\ 0.2,\ 0.0,\ 0.0918]$. Utilizing the S-approximation method, all the components times $0.8/0.8914$, so that $\overline{T}$ is adjusted to T^*=[0.4564, 0.6359, 0.8, 0.2615, 0.1795, 0.1795], and the corresponding $\overline{\tau} = [0.4508,\ 0.0918,\ 0.1795,\ 0.0,\ 0.0974]$.

When more thrusters experience thrust loss, the error rates for all methods increase compared to single-fault case [Figure 5]. As seen in the single-fault case, the faulty thrusters do not include T5 and T6. Therefore, the torque output error on the Heave and Roll axes remains zero through the T-approximation. However, both the T-approximation and S-approximation show large errors in direction, reaching 15%. The magnitude error also exceeds 15%. The magnitude error with PSO increases significantly, almost hitting 30%. In contrast, both GOA and SMA managed to adjust the submarine's direction effectively. SMA outperforms GOA in controlling magnitude error, keeping it within 5%. Even with two thrusters experiencing thrust loss, SMA's torque output optimization still shows good performance.

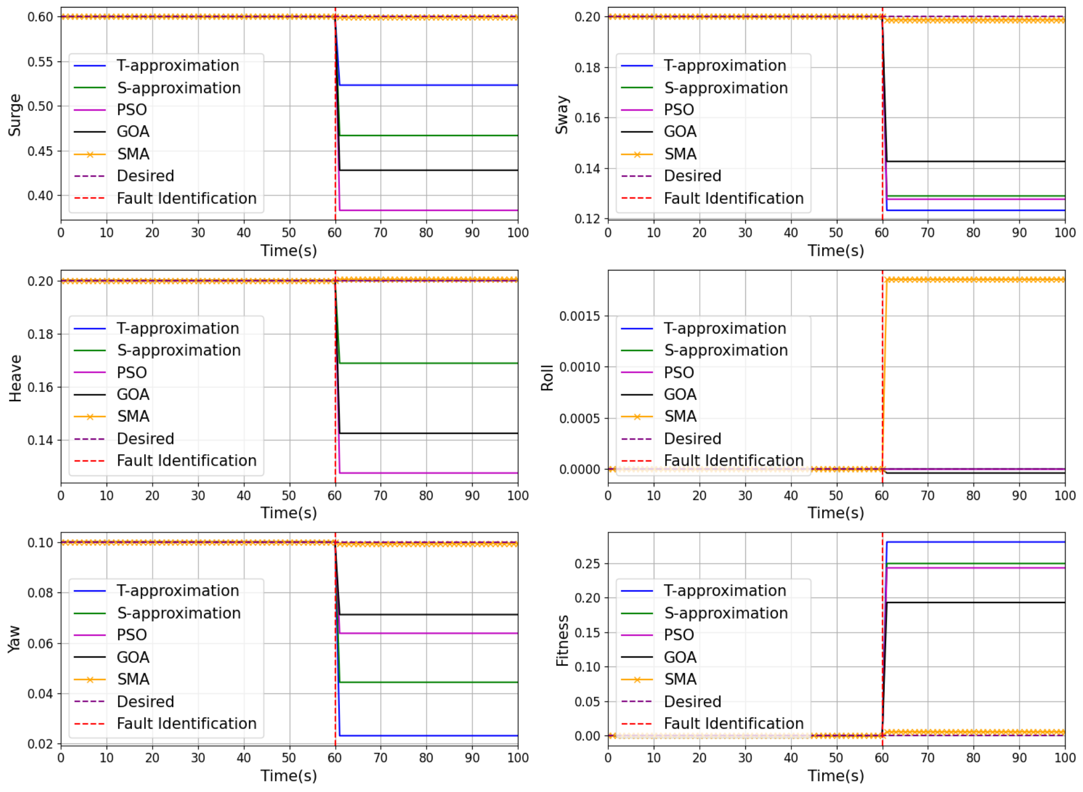

Figure 4. Multi-thruster reallocation problem for single-fault case.

Table 2. Results of double-fault case

	$\mathbf{T}^*$	$\bar{\tau}$	$\|\mathbf{e}\|$	$\|\theta_\mathbf{e}\|$
T-approximation	[0.5086, 0.7000, 0.8000, 0.2914, 0.2000, 0.2000]	[0.4825, 0.0825, 0.2000, 0.0000, 0.0918]	0.1664	0.1642
S-approximation	[0.4564, 0.6359, 0.8000, 0.2615, 0.1795, 0.1795]	[0.4508, 0.0918, 0.1795, 0.0000, 0.0974]	0.1855	0.1372
PSO	[0.1309, 0.2449, 0.6754, 0.3289, 0.1096, 0.1121]	[0.3450, 0.1151, 0.1108, −0.0013, 0.0581]	0.2863	0.0110
GOA	[0.3051, 0.4630, 0.8000, 0.3264, 0.1579, 0.1579]	[0.4736, 0.1579, 0.1579, 0.0000, 0.0789]	0.1413	0.0001
SMA	**[0.5134, 0.7000, 0.8000, 0.2409, 0.1879, 0.1888]**	**[0.5636, 0.1864, 0.1883, −0.0004, 0.0931]**	**0.0411**	**0.0027**

PSO: Particle swarm optimization; GOA: grasshopper optimization algorithm; SMA: slime mold algo-rithm.

4.3. Quadruple-fault case

The desired torque vector $\overline{\tau_d}$ is also assigned as [0.6, 0.2, 0.2, 0.0, 0.1]. At the 60-second mark, a catastrophic failure has significantly impacted the power distribution among all four thrusters. Specifically, the power output of $T1$ is reduced by 50%, $T2$ by 30%, $T3$ by 20%. The situation is more critical for T5, which has undergone a substantial 80% decrease in power, the corresponding W = [0.5, 0.7, 0.8, 1.0, 0.2, 1.0]. This drastic power imbalance poses a formidable challenge to the overall functionality and stability of the vehicle. The obtained calculation results are shown in Table 3.

The pseudo-inverse solution $\overline{T}$ is [0.5933, 0.7933, 0.8067, 0.2067, 0.2, 0.2] and three components are unfeasible because $\overline{T}_1 > 0.5, \overline{T}_2 > 0.7, \overline{T}_3 > 0.8$. Utilizing the T-approximation method, values exceeding control constraints are truncated. Specifically, 0.5933 will be set to 0.5, and 0.7933 will be set to 0.7, 0.8067 will be set to 0.8, so that $\overline{T}$ is adjusted to T^* = [0.5, 0.7, 0.8, 0.2067, 0.2, 0.2], and the corresponding $\overline{\tau}$=[0.3967, 0.1683, 0.12, -0.08, 0.0483]. Utilizing the S-approximation method, all the components times 0.5/0.5933, so that $\overline{T}$ is adjusted to T^*=[0.5, 0.6685, 0.6798, 0.1742, 0.1685, 0.1685], and the corresponding

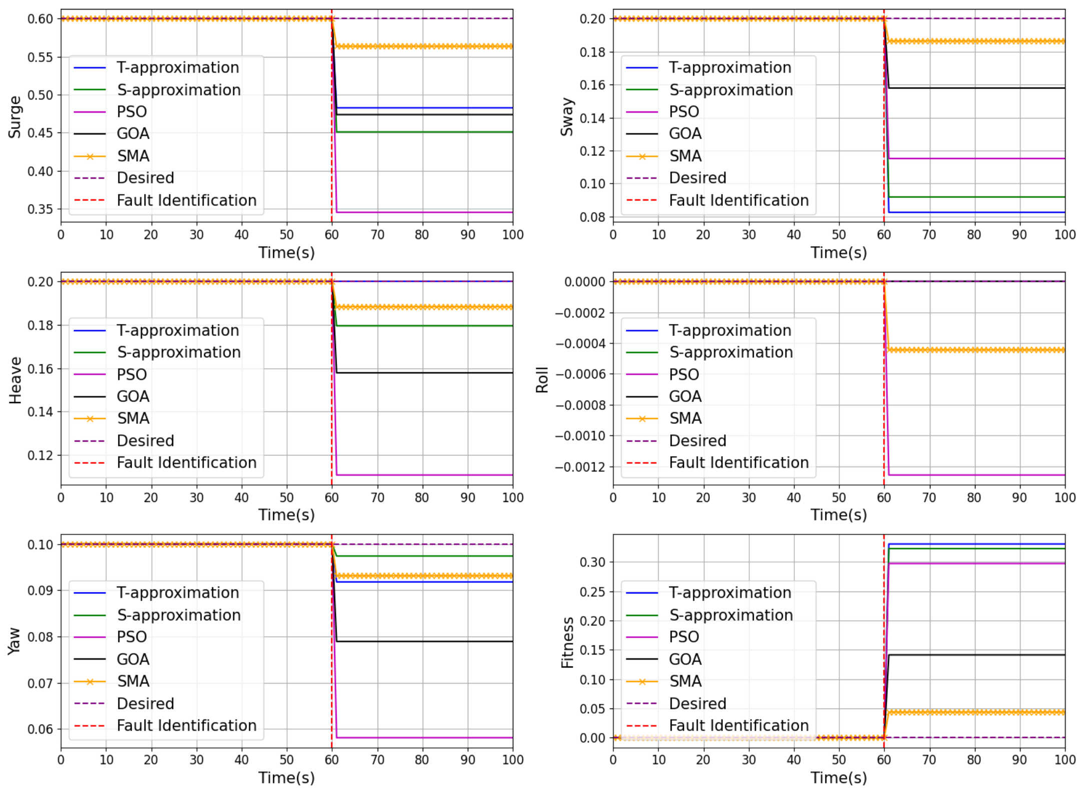

Figure 5. Multi-thruster reallocation problem for double-fault case.

Table 3. Results of quadruple-fault case

	$\mathbf{T}^*$	$\overline{\tau}$	$\|\mathbf{e}\|$	$\|\theta_{\mathbf{e}}\|$
T-approximation	$[0.5000, 0.7000, 0.8000, 0.2067, 0.2000, 0.2000]$	$[0.3967, 0.1683, 0.1200, -0.0800, 0.0483]$	0.2405	0.1989
S-approximation	$[0.5000, 0.6685, 0.6798, 0.1742, 0.1685, 0.1685]$	$[0.3590, 0.1469, 0.1011, -0.0674, 0.0379]$	0.2812	0.1927
PSO	$[0.1718, 0.3097, 0.7937, 0.3799, 0.1379, 0.1379]$	$[0.4138, 0.1379, 0.1379, 0.0000, 0.0690]$	0.2081	0.0001
GOA	$[-0.1937, -0.1123, 0.7637, 0.5194, 0.0814, 0.0814]$	$[0.2443, 0.0814, 0.0814, 0.0000, 0.0407]$	0.3977	0.0001
SMA	$\mathbf{[0.5000, 0.6878, 0.7999, 0.2444, 0.1858, 0.1857]}$	$\mathbf{[0.5580, 0.1858, 0.1858, 0.0000, 0.0919]}$	**0.0473**	**0.0017**

PSO: Particle swarm optimization; GOA: grasshopper optimization algorithm; SMA: slime mold algo-rithm.

$\overline{\tau} = [0.359,\ 0.1469,\ 0.1011,\ -0.0674,\ 0.0379].$

Due to damage to the vertical thruster, both T-approximation and S-approximation lost control over the Heave and Roll axes, leading to increased errors and loss of effective submarine control [Figure 6]. Despite these challenges, PSO, GOA, and SMA corrected the direction error successfully. Notably, even in such difficult conditions, SMA managed to keep the magnitude error within 5%. This control strategy maintained the submarine's direction without reducing its operational efficiency. SMA consistently delivered the best results in cases involving single-fault, double-fault, or quadruple-fault, demonstrating its potential as a reliable FTC method.

4.4. Efficiency and stability

This section compares the efficiency of the three algorithms: PSO, GOA, and SMA. Figure 7 illustrates the relationship between time and optimum fitness for three algorithms applied to thruster power redistribution under single-fault, double-fault, and quadruple-fault cases. Notably, after a large number of experiments, we

Table 4. Fitness and time of PSO, GOA, FTC-SMA in different fault cases

	Single-fault case		Double-fault case		Quadruple-fault case	
	Fitness	**Time (s)**	**Fitness**	**Time (s)**	**Fitness**	**Time (s)**
PSO	0.2429	0.1112	0.2973	0.1183	0.2082	0.1191
GOA	0.1928	0.2809	0.1413	0.2711	0.3978	0.2977
SMA	**0.0058**	**0.2370**	**0.0439**	**0.2353**	**0.0490**	**0.2480**

PSO: Particle swarm optimization; GOA: grasshopper opti-mization algorithm; FTC-SMA: fault-tolerant control strat-egy based on the slime mold algorithm.

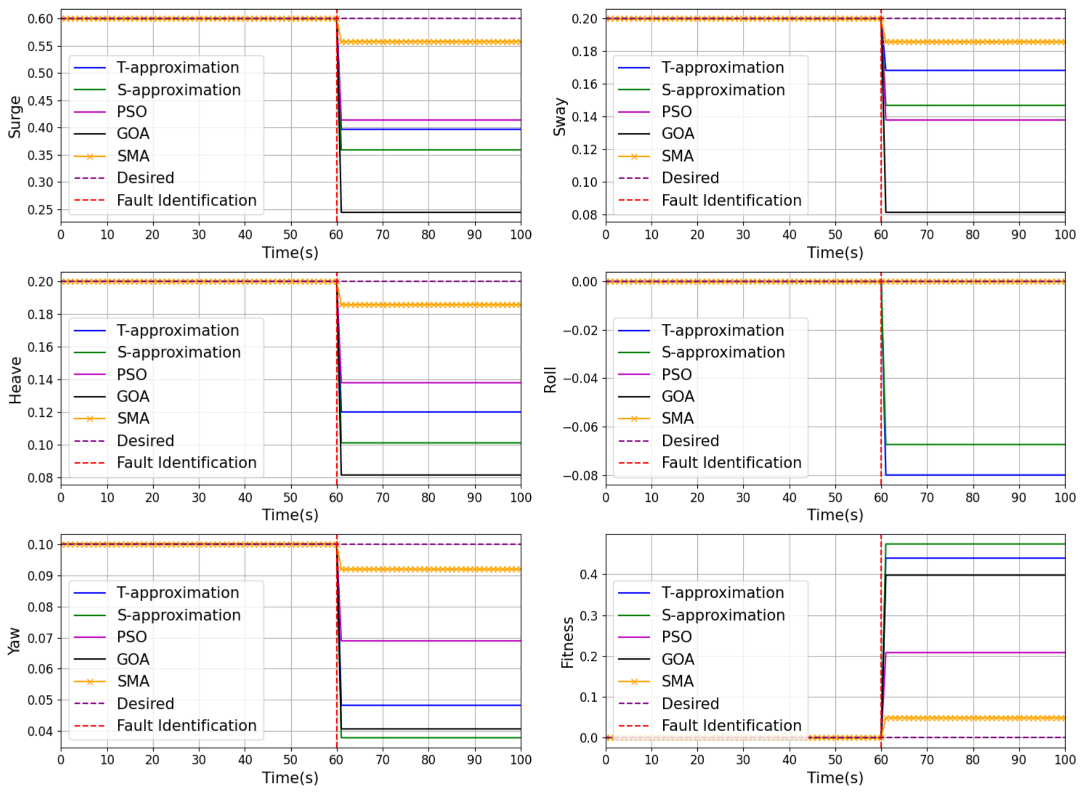

Figure 6. Multi-thruster reallocation problem for quadruple-fault case.

found that the increase in the number of thruster failures does not significantly affect the running time of each algorithm. Therefore, the running times shown in the paper are the average results obtained from multiple times.

The results are displayed in Table 4. All operating parameters are consistent. Runtime represents the total duration required for the algorithm to complete 200 iterations, ensuring that all algorithms have fully converged after these iterations. Due to its simple structure, PSO achieves the fastest running speed at just 0.119 s. However, PSO consistently falls short in terms of accuracy across all fault conditions. GOA, on the other hand, exhibits strong performance in accuracy for both single and double-fault cases. Despite this, its slower speed, approximately 0.3 s, fails to meet the real-time requirements for FTC systems. Moreover, under the quadruple-fault case, the performance of GOA deteriorates significantly; a sharp increase in magnitude error makes it incapable of FTC. SMA may not match the speed of PSO, with a running time of 0.24 s, but it consistently delivers the best results across all cases, including extreme ones. These observations affirm the effectiveness of FTC in UUVs and underscore the superior adaptability of SMA under various fault cases.

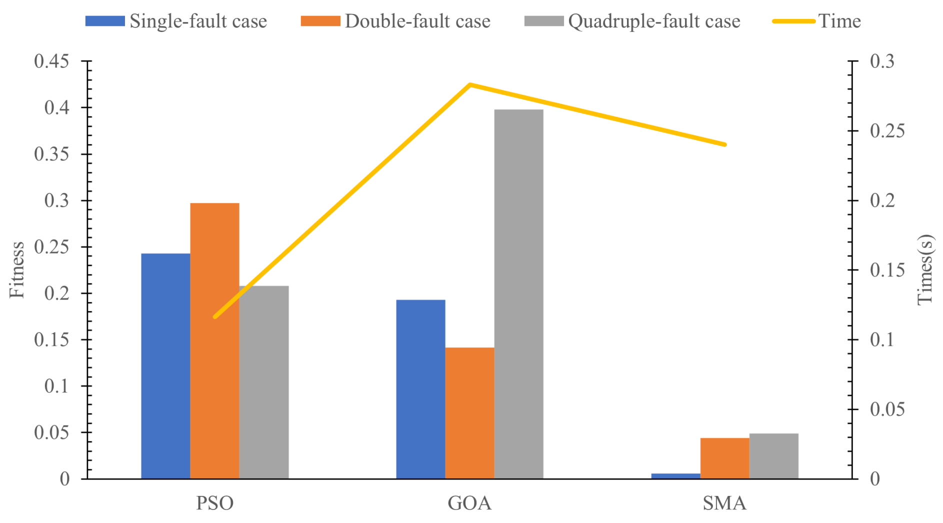

Figure 7. Runtime and fitness of PSO, GOA and SMA methods using the same operating parameters for different fault cases. PSO: Particle swarm optimization; GOA: grasshopper optimization algorithm; SMA: slime mold algorithm.

5. CONCLUSION

This paper employs an effective FTC-SMA. The FTC strategy leverages the SMA for rapid redistribution of thruster power, effectively constraining dynamic output within the limits of physical constraints. Simulation experiments were conducted using the weighted pseudo-inverse algorithm, PSO, GOA, and SMA for single-fault, double-fault, and quadruple-fault cases. The comparative analysis reveals that the control strategy based on SMA consistently yields superior results across various fault conditions. This conclusion is reinforced by the comprehensive comparison of the efficiency of PSO, GOA, and SMA under different fault conditions. In the future, pool experiments are planned to further evaluate the performance of the FTC-SMA algorithm in actual operations.

DECLARATIONS

Authors' contributions

Wrote the PSO, GOA, and SMA programs and the draft: Zeng, T.
Organized the experiment and guided the work: Zhu, D.; Gu, C.; Simon, X. Y.

Availability of data and materials

Not applicable.

Financial support and sponsorship

None.

Conflicts of interest

Simon, X. Y. is the Editor-in-Chief of the journal *Intelligence & Robotics*, Zhu, D. is the guest editor of the Special Issue of "Updates in Underwater Robotics", and Editorial Board Member of the journal *Intelligence & Robotics*, while the other authors have declared that they have no conflicts of interest.

Ethical approval and consent to participate

Not applicable.

Consent for publication
Not applicable.

Copyright

REFERENCES

1. Li, J.; Yang, G. Development and prospect of adaptive fault-tolerant control. *Control Decis.* **2014**, *29*, 1921-26. https://caod.oriprobe.com/articles/43182376/Development_and_prospect_of_adaptive_fault_tolerant_control.htm. (accessed on 2025-03-19)
2. Li, Y.; Hou, Y.; Liu, C. Adaptive optimal fault-tolerant control based on fault degree. *J. Comput. Eng. Appl.* **2021**, *57*, 295-302. DOI
3. Shen, Q.; Jiang, B.; Shi, P.; Lim, C. C. Novel neural networks-based fault tolerant control scheme with fault alarm. *IEEE Trans. Cybern.* **2014**, *44*, 2190-201. DOI
4. Shi, P.; Wang, X.; Meng, X.; He, M.; Mao, Y.; Wang, Z. Adaptive fault-tolerant control for open-circuit faults in dual three-phase PMSM drives. *IEEE Trans. Power Electron.* **2023**, *38*, 3676-88. DOI
5. Kadiyam, J.; Parashar, A.; Mohan, S.; Deshmukh, D. Actuator fault-tolerant control study of an underwater robot with four rotatable thrusters. *Ocean Eng.* **2020**, *197*, 106929. DOI
6. Baraniuk, T.; Simoni, R.; Weihmann, L. Fault-tolerant architecture for AUVs. In: *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, Porto, Portugal. Nov 06-09, 2018. IEEE, 2018. pp. 1-6. DOI
7. Panda, J. P.; Mitra, A.; Warrior, H. V. A review on the hydrodynamic characteristics of autonomous underwater vehicles. *J. Eng. Marit. Environ.* **2021**, *235*, 15-29. https://www.scribd.com/document/663456109/A-review-on-the-hydrodynamic-characteristics-of-autonomous-underwater-vehicles. (accessed on 2025-03-19)
8. Yu, J.; Zhang, A.; Wang, X. Research on thruster fault tolerant control allocation of a 7000m manned submarine. *Robot* **2006**, *28*, 519-24. https://robot.sia.cn/en/article/id/697. (accessed on 2025-03-19)
9. Esna, A. A.; Khaki, S. A.; Yazdanpanah, M. J. Reconfigurable control system design using eigen structure assignment: static, dynamic and robust approaches. *Int. J. Control* **2005**, *78*, 1005-16. DOI
10. Lin, C. M.; Chen, C. H. Robust fault-tolerant control for a biped robot using a recurrent cerebellar model articulation controller. *IEEE Trans. Syst. Man Cybern. B* **2007**, *37*, 110-23. DOI
11. Conte, G. Underwater robots: motion and force control of vehicle–manipulator systems. *Int. J. Adapt. Control Signal Process.* **2004**, *18*, 603-4. DOI
12. Yang, K. C.; Yuh, J.; Choi, S. K. Experimental study of fault-tolerant system design for underwater robots. In: *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, Leuven, Belgium. May 20, 1998. IEEE, 1998; pp. 1051-6. DOI
13. Yang, K. C. H.; Yuh, J.; Choi, S. K. Fault-tolerant system design of an autonomous underwater vehicle ODIN: an experimental study. *Int. J. Syst. Sci.* **1999**, *30*, 1011-9. DOI
14. Zhang, Y.; Jiang, J. Bibliographical review on reconfigurable fault-tolerant control systems. *Annu. Rev. Control* **2008**, *32*, 229-52. DOI
15. Podder, T. K.; Sarkar, N. Fault tolerant decomposition of thruster forces of an autonomous underwater vehicle. In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, Detroit, USA. May 10-15, 1999. IEEE, 1999; pp. 84-9. DOI
16. Podder, T. K.; Sarkar, N. Fault-tolerant control of an autonomous underwater vehicle under thruster redundancy. *Robotics Auton. Syst.* **2001**, *34*, 39-52. DOI
17. Fossen, T. I. Guidance and control of ocean vehicles. 1994. https://books.google.com/books/about/Guidance_and_Control_of_Ocean_Vehicles.html?id=cwJUAAAAMAAJ. (accessed on 2025-03-19)
18. Omerdic, E.; Roberts, G. Thruster fault diagnosis and accommodation for open-frame underwater vehicles. *Control Eng. Pract.* **2004**, *12*, 1575-98. DOI
19. Zhao, W.; Liu, H.; Wan, Y. Data-driven fault-tolerant formation control for nonlinear quadrotors under multiple simultaneous actuator faults. *Syst. Control Lett.* **2021**, *158*, 229-52. DOI
20. Chen, H.; Jiao, S.; Heidari, A. A.; Wang, M.; Chen, X.; Zhao, X. An opposition-based sine cosine approach with local search for parameter estimation of photovoltaic models. *Energy Convers. Manag.* **2019**, *195*, 927-42. DOI
21. Chen, H.; Xu, Y.; Wang, M.; Zhao, X. A balanced whale optimization algorithm for constrained engineering design problems. *Appl. Math. Model.* **2019**, *71*, 45-59. DOI
22. Wang, M.; Chen, H.; Yang, B.; et al. Toward an optimal kernel extreme learning machine using a chaotic moth-flame optimization strategy with applications in medical diagnoses. *Neurocomputing* **2017**, *267*, 69-84. DOI
23. Wang, M.; Chen, H. Chaotic multi-swarm whale optimizer boosted support vector machine for medical diagnosis. *Appl. Soft Comput.* **2020**, *88*, 105946. DOI
24. Chen, M.; Liu, Y.; Zhu, D.; Shen, A.; Wang, C.; Ji, K. Parameter identification of an open-frame underwater vehicle based on numerical simulation and quantum particle swarm optimization. *Intell. Robot.* **2024**, *4*, 216-29. DOI
25. Zhu, D.; Liu, J.; Yang, S. X. Particle swarm optimization approach to thruster fault-tolerant control of unmanned underwater vehicles. *Int. J. Robot. Autom.* **2011**, *26*. DOI

26. Tian, Q.; Wang, T.; Liu, B.; Ran, G. Thruster fault diagnostics and fault tolerant control for autonomous underwater vehicle with ocean currents. *Machines* **2022**, *10*, 582. DOI
27. Zhu, D.; Liu, Q.; Hu, Z. Fault-tolerant control algorithm of the manned submarine with multi-thruster based on quantum-behaved particle swarm optimization. *Int. J. Control* **2011**, *84*, 1817-29. DOI
28. Zhu, D. J.; Wang, L.; Hu, Z.; Yang, S. X. A grasshopper optimization-based fault-tolerant control algorithm for a human occupied submarine with the multi-thruster system. *Ocean Eng.* **2021**, *242*, 110101. DOI
29. Zhu, D. J.; Wang, L.; Zhang, H.; Yang, S. X. A GOA-based fault-tolerant trajectory tracking control for an underwater vehicle of multi-thruster system without actuator saturation. *IEEE Trans. Autom. Sci. Eng.* **2024**, *21*, 771-82. DOI
30. Sun, B.; Pang, W.; Chen, M.; Zhu, D. Development and experimental verification of search and rescue ROV. *Intell. Robot.* **2022**, *4*, 355-70. DOI
31. Li, S.; Chen, H.; Wang, M.; Heidari, A. A.; Mirjalili, S. Slime mould algorithm: a new method for stochastic optimization. *Future Gener. Comput. Syst.* **2020**, *111*, 300-23. DOI