Joint Trajectory Replanning for Mars Ascent Vehicle Under Propulsion System Faults: A Suboptimal Learning-Based Warm-Start Approach

Kun Li, Guangtao Ran, *Member, IEEE*, Yanning Guo, Ju H. Park, *Senior Member, IEEE*, and Yao Zhang, *Senior Member, IEEE*

***Abstract*—This article presents a suboptimal joint trajectory replanning (SJTR) method for Mars ascent vehicle (MAV) launch missions under propulsion system faults. Conventional step-by-step trajectory replanning may fail to make timely decisions, risking mission failure. The SJTR method formulates a joint convex optimization problem of target orbit and flight trajectory after a fault. By applying penalty coefficients for terminal constraints, it adheres to the orbit redecision principles, enabling a concise and rapid solution. To further enhance the convergence and the accuracy of orbit-type determination, a learning-based warm-start scheme is proposed. Offline, a deep neural network (DNN) is trained with data generated by various trajectory replanning methods following the redecision principles. Online, the DNN provides initial guesses for the time optimization variables based on the fault scenario. Numerical simulations on mass flow rate and specific impulse drops validate the reliability of the proposed method, demonstrating at least 49.5% higher computational efficiency compared with the upgrading and downgrading replanning methods.**

***Index Terms*—Deep neural network (DNN), Mars ascent vehicle (MAV), propulsion system faults, trajectory replanning.**

I. Introduction

For future Mars sample return and crewed return missions, the design of the Mars ascent vehicle (MAV) is a crucial and challenging aspect. On one hand, the Martian environment has significant uncertainties, such as the impact of Martian winds and solar storms [1]. On the other hand, the temperature conditions on Mars are harsh, and unexpected engine and propellant issues may arise after ignition. Recent experiences indicate that rocket faults occur frequently, even without considering adverse environmental conditions, with propulsion system faults being the most common issue [2]. When the system fault is minor, the MAV can reach the target orbit using classical closed-loop guidance methods. However, when the fault is so severe that the original target orbit cannot be reached, traditional guidance methods alone are far from sufficient. In such cases, if the MAV lacks redundancy and contingency plans, there is a risk of mission failure.

Received 11 September 2024; revised 5 March 2025 and 23 June 2025; accepted 8 August 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62403166, Grant 62273118, and Grant 62473110; in part by the Natural Science Foundation of Heilongjiang Province under Grant LH2022F023; and in part by the Young Elite Scientists Sponsorship Program by the China Association for Science and Technology (CAST) under Grant YESS20240112. The work of Ju H. Park was supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science and Information and Communications Technology (MSIT) under Grant RS-2019-NR040065. *(Corresponding author: Yanning Guo.)*

Kun Li, Guangtao Ran, and Yanning Guo are with the Department of Control Science and Engineering, Harbin Institute of Technology, Harbin 150001, China (e-mail: lkun005@stu.hit.edu.cn; ranguangtao@hit.edu.cn; guoyn@hit.edu.cn).

Ju H. Park is with the Department of Electrical Engineering, Yeungnam University, Gyeongsan 38541, South Korea (e-mail: jessie@ynu.ac.kr).

Yao Zhang is with the Department of Mechanical Engineering, University College London, WC1E 7JE London, U.K. (e-mail: yao.zhang@ucl.ac.uk).

Digital Object Identifier 10.1109/TNNLS.2025.3598120

Regardless of whether faults occur, the ultimate objective is to optimize flight trajectories to the target orbit. After significant development in recent years, convex optimization can be widely applied to solve many trajectory optimization problems [3], [4], [5], [6], [7]. It can find the optimal solution within a finite time, effectively meeting the high real-time requirements in some scenarios. Currently, methods such as lossless convexification and successive convexification are used to transform nonconvex and nonlinear terms into convex forms [8], [9]. Moreover, there is a growing number of methods to improve the performance of solving convex optimization problems combined with sequential convex programming (SCP) frameworks. Hong et al. [10] proposed a novel model predictive convex programming for a class of nonlinear systems with state and input constraints, addressing optimal guidance problems with terrain constraints. A combination of modified Chebyshev–Picard iteration (MCPI) and SCP is proposed to approximate dynamic equations, eliminating state variables in finite-dimensional subproblems and improving convergence performance [11], [12]. To ensure that linearized subproblems approximate the original problem well and prevent potential infeasibility, trust-region constraints and virtual control terms were introduced in [13] and validated in a Falcon 9 launch vehicle simulation scenario.

When typical propulsion system faults such as mass flow rate drop and specific impulse drop occur, if the flight still follows the reference commands and trajectory, the fuel may be consumed prematurely. Therefore, it is necessary to redecide the target orbit and optimize the flight trajectory after detecting a fault using fault diagnosis methods [14], [15], [16]. Even if the original target orbit cannot be reached, entering an appropriate rescue orbit for potential rescue operations helps minimize losses. For example, Song et al. [17]

studied an autonomous mission reconstruction technology that simulates various failure scenarios occurring during missions. The algorithm evaluates the remaining performance of the rocket and plans new objectives and corresponding flight paths through iterative guidance mode or segmented state triggered method, enabling the rocket to enter the target orbit as expected or deploy the payload in other degraded orbits. This effectively avoids the risk of total loss in the face of such failures [17]. Ma et al. [18] proposed an improved parallel-structured Newton-type guidance algorithm, which can be applied to problems with free-time and path constraints, effectively solving the problem that the thrust drop may lead to the failure of the launch vehicles during endoatmospheric flight. Trajectory replanning after fault essentially involves two parts: redecision of the target orbit and optimization of the flight trajectory. State-triggered indicator (STI) method correlates different rescue orbits with transition states defined by three indicators, addressing the online optimization problem of target orbit and flight trajectory after thrust drop faults [19]. Miao et al. [20] introduced an auxiliary phases' method that avoids linearization of the objective function and terminal constraints, enhancing the convergence performance of trajectory replanning.

In recent years, deep neural networks (DNNs) have made significant strides in solving trajectory planning problems [21], [22], [23]. Chai et al. [24] proposed a comprehensive trajectory planning approach for solving the reentry flight of hypersonic vehicles with six degrees of freedom, adopting a two-layer structure that combines desensitized trajectory optimization and DNN. Hua and Fang [25] proposed a learning-based trajectory generation framework for a quadrotor, which combines reinforcement learning and imitation learning to make human-like decisions online, ensuring real-time and practically reliable solution. Scholars have also conducted some research on using DNN to assist in trajectory replanning after failures. He et al. [26] proposed a DNN-based adaptive collocation method by establishing mappings between offline fault situations and optimal rescue orbits and terminal control variables, which improves the efficiency of online trajectory replanning. Moreover, the concept of dynamic multiobjective optimization may inspire new approaches to solving the MAV trajectory replanning problem [27], [28].

Ensuring feasibility and convergence in each optimization during trajectory replanning is crucial [29]. A well-set initial guess can reduce infeasible situations and improve solution efficiency. To address complex ascent problems, a three-step continuation scheme is devised to enhance the solution success rate using the solutions from simplified problems as initial guesses for the real problems [13]. Banerjee et al. [30] introduced a novel approach of using outputs from trained models to warm-start nonlinear solvers, reducing computational time while obtaining feasible and locally optimal solutions of trajectory optimization problems.

To the best of the authors' knowledge, very few studies are currently available on MAV trajectory replanning after faults. This is a multiphase, highly nonlinear, free-terminal time optimization problem that is inherently difficult to solve, requiring algorithms with high real-time capability. Unlike typical ascent trajectory optimization problems where the target orbit is usually predetermined, in this case, the target orbit needs to be optimized. Although the existing joint optimization methods can solve this problem, they still cannot avoid multistep judgment and optimization [19], [20]. Moreover, the search space for solutions is large, placing high demands on convergence of optimization algorithms. Directly using neural network mappings to derive trajectories toward the target may also encounter several issues. For example, neural networks rely on training data, and deviations from expected trajectories or uncertainties in the actual model may lead to significant deviations. Another limitation is that this approach is only applicable to a limited number of fault modes, and for unforeseen severe faults, infeasible situations may occur. In addition, it is challenging to provide reasonable initial guesses, which hinders the convergence of the algorithm and computational efficiency [31], [32].

The main contributions of this article are summarized as follows.

1) A concise, fast, and reliable suboptimal joint trajectory replanning (SJTR) method is proposed to solve the MAV trajectory replanning problem after faults. It eliminates the need for separate decision-making on target orbits and trajectory optimization during flight. The solution to this joint trajectory replanning problem can be obtained directly instead of using STIs for step-by-step judgment and optimization.
2) A learning-based warm-start method is designed, which provides a reasonable initial guess for the SJTR method through an offline trained neural network, avoiding infeasible situations and improving solution efficiency. In addition, it addresses the issue of inaccurate orbit type determination that can occur in some edge-case faults of the SJTR method, thereby enhancing reliability.

The rest of this article will be organized as follows. Section II introduces the dynamics, establishing the trajectory optimization problem and target orbit redecision principles. Section III analyzes the general trajectory replanning method, the SJTR method, and the learning-based warm-start scheme. Detailed simulation results and comparative studies are presented in Section IV. Finally, Section V concludes this article.

II. Problem Formulation

A. Symbols' Appointment

Adopted symbols in this article are summarized in Table I.

B. Dynamics Under Faults

The whole flight process can be divided into three phases which are the ascending phase, the coasting phase, and the orbiting phase. The characteristics of each phase are as follows.

1) *Ascending Phase:* The MAV takes off with thrust provided by the first-stage engines, and this phase stops when the first-stage propellant is completely consumed.
2) *Coasting Phase:* No thrust is applied throughout the phase, and the end time of this phase is flexible.

TABLE I
SUMMARY OF ADOPTED SYMBOLS

Symbol	Description	Unit
D	Aerodynamic drag	N
C_d	Drag coefficient	-
S	Reference area of the MAV	m^2
ρ	Atmospheric density	$\mathrm{kg/m}^3$
$\boldsymbol{v}_{\mathrm{rel}}$	Velocity vector relative to Mars	m/s
$\boldsymbol{r}$, $\boldsymbol{v}$	Position and velocity vectors	m and m/s
$\boldsymbol{\omega}$	Angular velocity of Mars	rad/s
h	Altitude from the surface of Mars	m
h_0	Martian density scale height	m
ρ_0	Atmospheric density at sea level	$\mathrm{kg/m}^3$
L	Variable distinguishing the fault mode	-
μ	Martian gravitational parameter	$\mathrm{m}^3/\mathrm{s}^2$
g_0	Standard gravity	$\mathrm{m/s}^2$
I_{sp}	Specific impulse of the engine	s
m	Mass of the MAV	kg
t_{fail}	Time of fault occurrence	s
T	Thrust magnitude	N
$\boldsymbol{u}$	Thrust direction vector	-
η, κ	Variables determined by the fault mode	-
$\boldsymbol{x}$	State vector (position, velocity, mass)	-
a^*, e^*, i^*, Ω^*	Target orbital elements (semi-major axis, eccentricity, inclination, longitude of ascending node)	-
$\mathbf{1}_h$	Unit direction vector of the orbital angular momentum	-
J	Cost function	-
a_f	Semi-major axis at the final time	m
a_{safe}	Minimum safe orbital altitude	m
$\lambda_i, \lambda_\Omega$	Weight coefficients	-
$\boldsymbol{\Delta}_\phi$	Slack variable	-
$\boldsymbol{\omega}_\phi$	Penalty parameter	-
$\boldsymbol{\Lambda}$	Vector of optimization variables	-
$\boldsymbol{\delta}$	Trust-region radius	-
$\boldsymbol{\alpha}$, $\boldsymbol{\beta}$	Input and output vectors for the DNN	-
$\boldsymbol{x}_{\mathrm{fail}}$	State vector at the time of fault	-
R_M	Radius of Mars	m

3) *Orbiting Phase:* The first stage separates, and propulsion is provided by the second stage engine. This phase ends when the target orbit constraints are satisfied.

Since aerodynamic lift is negligible compared with thrust, aerodynamic drag is considered the primary form of aerodynamic force. The definition of aerodynamic drag is as follows:

$$D = C_d S \rho \|\boldsymbol{v}_{\mathrm{rel}}\|^2/2 \tag{1}$$

where $\boldsymbol{v}_{\mathrm{rel}} = \boldsymbol{v} - \boldsymbol{\omega} \times \boldsymbol{r}$ is the velocity vector relative to Mars, C_d is the drag coefficient, S is the MAV's reference area, $\boldsymbol{\omega}$ is the angular velocity of the Mars, and $\boldsymbol{r} = [r_x, r_y, r_z]^\top$ and $\boldsymbol{v} = [v_x, v_y, v_z]^\top$ denote the inertial position and the velocity vectors of the MAV, respectively. The atmosphere density is considered only within an altitude of 120 km, which is modeled as $\rho(t) = \rho_0 \exp(-h(t)/h_0)$, where h denotes the altitude from the surface of Mars, h_0 is the Martian density scale height, and ρ_0 is the atmospheric density at sea level.

We focus on the problem of thrust drop faults occurring during the ascending phase, which is the most common type of propulsion system faults. Thrust drop faults can be mainly divided into two cases: mass flow rate drop ($L = 0$) and specific impulse drop ($L = 1$), where L is the variable that distinguishes the fault mode. The essential difference between these two fault modes lies in whether the propellant consumption changes while the thrust drops [33]. When a blockage fault occurs, it usually corresponds to the mass flow rate drop fault. In this case, the fuel consumption rate decreases accordingly. In the Mars ascent problem with a thin atmosphere, the impact of this type of fault is generally not fatal. When a leakage fault occurs, it usually corresponds to the specific impulse drop fault. In this situation, the propellant consumption rate remains unchanged, but the thrust magnitude decreases and not all the propellant is converted to thrust, which represents a worse scenario.

Considering the standard flight dynamics model of the MAV, in the Martian centered inertial (MCI) coordinate system, we establish a unified three-degree-of-freedom dynamics equation that takes thrust drop faults into account

$$\dot{\boldsymbol{r}} = \boldsymbol{v} \tag{2}$$

$$\dot{\boldsymbol{v}} = -\frac{\mu}{\|\boldsymbol{r}\|^3}\boldsymbol{r} + \frac{\eta T}{m}\boldsymbol{u} - \frac{D}{m\|\boldsymbol{v}_{\mathrm{rel}}\|}\boldsymbol{v}_{\mathrm{rel}} \tag{3}$$

$$\dot{m} = -\frac{\kappa T}{I_{\mathrm{sp}} g_0} \tag{4}$$

where μ is the Martian gravitational parameter, g_0 is the standard gravity, I_{sp} is the specific impulse of the engine, and m denotes the mass of the MAV. In addition, the thrust magnitude T of the MAV is fixed, with the only adjustable control variable being the thrust direction vector $\boldsymbol{u} = [u_x, u_y, u_z]^\top$. The variables η and κ are determined by the following piecewise function:

$$(\eta, \kappa) = \begin{cases} (1, 1), & t < t_{\mathrm{fail}} \\ (\eta^*, 1), & t \geq t_{\mathrm{fail}} \text{ and } L = 1 \\ (\eta^*, \eta^*), & t \geq t_{\mathrm{fail}} \text{ and } L = 0 \end{cases} \tag{5}$$

where $0 \leq \eta^* < 1$ is the percentage of remaining thrust, and t and t_{fail} are the flight time and the time of fault occurrence, respectively.

Dealing with the situation after faults is a complex issue that involves processes such as fault prediction, fault diagnosis, and fault-tolerant control [34], [35]. These are not within the scope of discussion in this article. For the sake of simplicity in analysis, we make the following assumptions.

Assumption 1: The fault occurs only once, and once the fault happens, its magnitude will not change during the process.

Assumption 2: The system can accurately and in real-time detect the occurrence time and situations of the fault.

C. Trajectory Optimization Problem

In this section, we provide the mathematical description of the basic fuel-optimal trajectory optimization problem.

To handle the multiphase problems in a unified manner, we connect each phase with the linkage constraints

$$\boldsymbol{r}_{p,0} = \boldsymbol{r}_{p-1,f}, \ \boldsymbol{v}_{p,0} = \boldsymbol{v}_{p-1,f}, \ m_{3,0} = m_{2,f} - m_{1,\mathrm{dry}} \tag{6}$$

where $p(p=2,3)$ is the phase number, $m_{1,\mathrm{dry}}$ denotes the dry mass of first stage, and $(\cdot)_{p,0}$ and $(\cdot)_{p,f}$ represent the state at the initial and final times of phase p, respectively.

The initial state constraint is as follows:

$$\boldsymbol{x}_{1,0} = \boldsymbol{x}_0 \tag{7}$$

where $\boldsymbol{x} = [\boldsymbol{r}^\top, \boldsymbol{v}^\top, m]^\top$ represents the state vector consisting of position, velocity, and mass.

The terminal constraints $\boldsymbol{\phi}(\boldsymbol{x}_{3,f})$ can be expressed by referring to the five-constraint problem in [36] as

$$\|\boldsymbol{r}_{3,f}\| - a^* = 0 \tag{8}$$

$$\|\boldsymbol{v}_{3,f}\| - \sqrt{\mu/a^*} = 0 \tag{9}$$

$$\boldsymbol{r}_{3,f}{}^\top \boldsymbol{v}_{3,f} = 0 \tag{10}$$

$$\boldsymbol{r}_{3,f}{}^\top \boldsymbol{1}_h = 0 \tag{11}$$

$$\boldsymbol{v}_{3,f}{}^\top \boldsymbol{1}_h = 0. \tag{12}$$

The unit direction vector of the orbital angular momentum is represented by $\boldsymbol{1}_h^{\mathrm{MCI}} = [\sin\Omega^* \sin i^*, -\cos\Omega^* \sin i^*, \cos i^*]^\top$, and $\boldsymbol{1}_h$ is the representation of $\boldsymbol{1}_h^{\mathrm{MCI}}$ in the inertial launch plumbline system. Using (8)–(12), we can constrain the semi-major axis a^*, eccentricity e^*, orbital inclination i^*, and longitude of ascending node Ω^* of the target orbit.

The final mass $m_{3,f}$ must be greater than the dry mass of the second stage $m_{2,\mathrm{dry}}$ and the mass of the payload m_{payload}

$$m_{3,f} \geq m_{2,\mathrm{dry}} + m_{\mathrm{payload}}. \tag{13}$$

The magnitude of the thrust direction vector is equal to one during the ascending and orbiting phases. This constraint is often transformed through lossless convexification [37]

$$\|\boldsymbol{u}_1\| \leq 1, \quad \|\boldsymbol{u}_3\| \leq 1. \tag{14}$$

The performance index is the final mass of the MAV and since the thrust magnitude is constant, the performance index can be equivalently expressed as

$$J_1 = t_{3,f} - t_{2,f} \tag{15}$$

where $t_{2,f}$ and $t_{3,f}$ are the final time of the coasting and orbiting phases, respectively.

In summary, the basic optimization problem can be described as P1.

P1:

Minimize: Equation (15).

Subject to: Equations (2)–(14).

D. Target Orbit Redecision Principles

Without considering faults, the target orbit is usually predetermined; however, when a thrust drop fault occurs, if the MAV still follows the original trajectory, there is a risk that the target orbit cannot be reached or result in mission failure. It is necessary not only to design an optimal flight trajectory but also to evaluate the capability to achieve the orbit according to the fault situation and the actual state of the MAV in advance, so as to complete the redecision of the target orbit.

The propellant consumption for adjusting the orbital plane is much larger than that for adjusting the orbital altitude. Referring to the definition of the STI, to ensure safety, we should prioritize the orbit altitude when redeciding the target orbit, and the adjustment of the orbital plane will be considered only when the remaining capacity is greater than a certain threshold. The target and rescue orbits studied in this article are circular orbits, and the specific types of entry into orbit corresponding to the redecision principles can be categorized into four types.

1) *Original Target Orbit:* Small fault condition, even if a thrust drop fault occurs, the remaining propellant is able to support the MAV into the original target orbit.
2) *Rescue Orbit Type I:* Medium fault condition, cannot enter the original target orbit, but can reach the same altitude as the target orbit. While maintaining its orbital altitude and further reducing the deviation of the orbital plane to make the rescue orbit as close as possible to the target orbit.
3) *Rescue Orbit Type II:* Large fault condition, cannot enter the original target orbit or reach the same altitude as the target orbit. At this time, only the deviation of the orbit altitude from the original target orbit altitude is reduced, without considering the orbital plane deviation.
4) *Mission Failure:* Severe fault condition, the MAV cannot reach the minimum safe orbit altitude.

III. Trajectory Replanning After Faults

Trajectory replanning after thrust drop fault is an optimal joint optimization problem of rescue orbit and flight trajectory. However, it is tough to solve these two coupled problems directly. This is because different fault scenarios correspond to different types of rescue orbits, making it challenging to represent the problem in a unified form. This may lead to nonconvergence of the optimization algorithm or low computational efficiency. Since the rescue orbit serves as the target orbit for flight trajectory optimization and is unknown, this greatly expands the search space for the optimal solution and makes it difficult to provide an idea initial guess.

A. General Trajectory Replanning Method

Drawing on the multistep optimization framework based on STI, this section uses the general trajectory replanning method to construct the procedure that follows the principle mentioned in Section II for solving the problem.

First, without considering the semi-major axis, orbital inclination, and longitude of ascending node constraints at the final time, (9) is changed to

$$\|\boldsymbol{v}_{3,f}\| - \sqrt{\mu/a_f} = 0 \tag{16}$$

where $a_f = \|\boldsymbol{r}_{3,f}\|$ is the magnitude of the semi-major axis at the final time. The optimization problem is then transformed into P2, which optimizes the highest circular orbit.

P2:

Minimize: $J_2 = -a_f$.

Subject to: Equations (2)–(6), (10), (13), (14), and (16).

When a_f is less than the safe orbit altitude a_{safe}, it indicates mission failure. When $a_{\mathrm{safe}} \leq a_f \leq a^*$, the optimized result is considered the highest circular rescue orbit, classified as

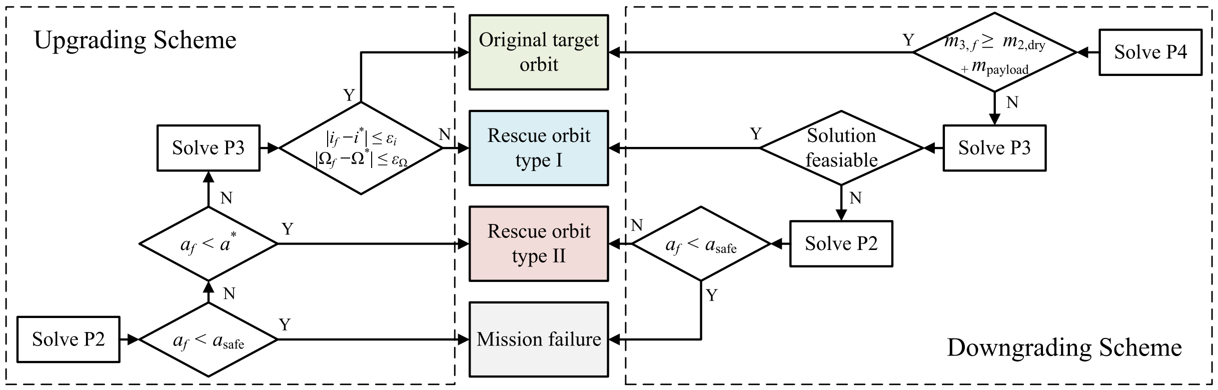


Fig. 1. General trajectory replanning method.

rescue orbit type II. If $a_f > a^*$, it means that the MAV can reach the original target orbit altitude while further reducing the deviation of the orbital inclination i_f and longitude of ascending node Ω_f. In this case, a new problem P3 needs to be solved, where it is necessary to ensure $a_f = a^*$. P3 is used to determine the rescue orbit that minimizes the orbital plane difference at the same altitude as the original target orbit.

P3:

Minimize: $J_3 = \lambda_i |i_f - i^*| + \lambda_\Omega |\Omega_f - \Omega^*|$.

Subject to: Equations (2)–(10) and (13) and (14).

In P3, λ_i and λ_Ω are the weights. This problem can simultaneously correspond to entering the original target orbit and the rescue orbit type I. When the performance index is small enough, it implies that the semi-major axis and the orbital plane elements both meet the requirements of the target orbit. Otherwise, it belongs to the rescue orbit type I.

Now, P2 and P3 have been formulated for the scenarios corresponding to the four types of orbit. The process of solving this issue involves initially considering the worst case scenario and then gradually "upgrade" the mission. In contrast, a method of "downgrading" is proposed to first consider the best-case scenario [20], [38]. Unlike the upgrading scheme, this downgrading method initially does not constrain the final mass of the MAV and its aim is to first obtain a set of solutions that satisfy all the target orbital constraints. The steps for the downgrading scheme are as follows. We define a new problem.

P4:

Minimize: $J_4 = -m_{3,f}$.

Subject to: Equations (2)–(12) and (14).

By solving P4, if the obtained solution satisfies (13), it indicates that the MAV has sufficient fuel to enter the original target orbit. Conversely, it means the MAV needs to consume more fuel than available, thus necessitating entry into a rescue orbit or resulting in mission failure. If entering a rescue orbit is required, the next step is to solve P3. If the problem is feasible, the resulting trajectory is for entering the rescue orbit type I. If the problem is infeasible, then determine whether it is possible to enter the rescue orbit type II by solving P2 to obtain the highest rescue orbit. If this highest rescue orbit exceeds the altitude of the safe orbit, then this is the solution for the rescue orbit type II. Otherwise, the mission is deemed a failure.

The general trajectory replanning method is shown in Fig. 1. It should be noted that the upgrading scheme that involves two optimization problems is not necessarily more time-saving than the downgrading scheme that involves three optimization problems. For example, when a minor fault occurs and it is possible to enter the original target orbit, the downgrading scheme only requires one judgment, while the upgrading scheme needs to run through all the steps. Therefore, the solving speed is affected by the fault situation and the order of judgment. In practical applications, the upgrading or downgrading strategies can be flexibly chosen based on the specific requirements.

B. SJTR Method

Due to the unknown rescue orbit, the search space for the optimal solution is enormous. Consequently, the general trajectory replanning method requires solving the optimization problem step-by-step, which may lead to time-consuming and infeasible solution propagation. In addition, the divergence in search directions caused by conflicts between the orbital plane and shape elements significantly reduces solution efficiency or rendering infeasible.

For typical trajectory optimization problems, precision and optimality are crucial. However, for the problem of trajectory replanning after faults studied in this article, convergence and computational efficiency are more important. MCPI is a method known for its good convergence and computational efficiency [39], [40]. Although the convergence of MCPI cannot be strictly proven when solving general nonlinear problems [41], many studies have shown that for ascent problems, the convergence bound of this method is quite large even without an ideal initial guess [12], [42]. In this section, we developed the SJTR method within the MCPI framework.

For solving an initial value problem of a nonlinear differential equation, we usually use the integral form for calculation

$$\boldsymbol{x}(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}(s, \boldsymbol{x}(s))\,\mathrm{d}s, \quad t \in [t_0, t_f]. \tag{17}$$

In contrast, MCPI offers a new way of solving the problem. The state trajectories are approximated by Picard iteration, and then continuously iterated to ensure their computational accuracy. The formula for Picard iteration is as follows:

$$\boldsymbol{x}^k(t) = \boldsymbol{x}(t_0) + \int_{t_0}^{t} \boldsymbol{f}\left(s, \boldsymbol{x}^{k-1}(s)\right) \mathrm{d}s, \quad k = 1, 2, \ldots \tag{18}$$

where $\boldsymbol{x}^k$ denotes the solution of the k^{th} iteration in the successive solution process. We perform $t = (t_f + t_0)/2 + (t_f - t_0)\tau/2$ to transform the time domain from $t \in [t_0, t_f]$ to $\tau \in [-1, 1]$ and obtain the new Picard iteration formula

$$\boldsymbol{x}^k(\tau) = \boldsymbol{x}_0 + \int_{-1}^{\tau} \boldsymbol{g}\left(s, \boldsymbol{x}^{k-1}(s)\right) \mathrm{d}s, \quad k = 1, 2, \ldots \tag{19}$$

Use N Chebyshev–Gauss–Lobatto nodes for discretization

$$\tau_j = -\cos(j\pi/N), \quad j = 0, 1, 2, \ldots, N. \tag{20}$$

We can get the approximated force function using orthogonal Chebyshev polynomials

$$\boldsymbol{g}\left(\tau, \boldsymbol{x}^{k-1}(\tau)\right) = \sum_{i=0}^{N-1} \boldsymbol{F}_i^{k-1} T_i(\tau) \tag{21}$$

where $T_i(\tau) = \cos(i \arccos(\tau))$ represents the basis function. Based on the discrete orthogonality of the Chebyshev polynomials, we can compute the coefficient vector $\boldsymbol{F}_i^{k-1} = (1/c_i)\sum_{j=0}^{N} z_j \boldsymbol{g}(\tau_j, \boldsymbol{x}^{k-1}(\tau_j)) T_i(\tau_j)$, $c_0 = N$; $c_i = N/2$, for $i = 1, 2, \ldots, N$; $z_0 = z_N = 1/2$; $z_j = 1$ for $j = 1, 2, \ldots, N-1$.

The Picard iteration formulas for velocity and position are as follows:

$$\boldsymbol{v}_p^{k+1}\left(\tau_p\right) \approx \boldsymbol{v}_{p,0} + \int_{-1}^{\tau_p} \frac{\left(t_{p,f} - t_{p,0}\right)}{2} \boldsymbol{f}_{v,p}\left(\boldsymbol{x}_p^k, \boldsymbol{u}_p\right) \mathrm{d}s \tag{22}$$

$$\boldsymbol{r}_p^{k+1}\left(\tau_p\right) = \boldsymbol{r}_{p,0} + \int_{-1}^{\tau_p} \frac{\left(t_{p,f} - t_{p,0}\right)}{2} \boldsymbol{v}_p^{k+1} \mathrm{d}s \tag{23}$$

where $t_{p,0}$ and $t_{p,f}$ are the initial and final times of phase p, respectively. $\boldsymbol{f}_{v,p}$ is the velocity parts of the function $\boldsymbol{f}$, $\boldsymbol{x}_p^k$ is the reference state trajectory of the kth iteration and $\boldsymbol{u}_p$ is the control history.

Using (22) and (23) to compute the state, we do not need to integrate the state variables. This means that when solving the optimization problem, there is no need to set complex dynamics constraints and the states at each step are not coupled with each other. Instead, we can recover it based on the result of the previous iteration and the control history using Picard iteration. This changes the optimization variables from $\boldsymbol{\Xi}_0$ $(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{u}_1, \boldsymbol{u}_3, t_{2,f}, t_{3,f})$ to $\boldsymbol{\Xi}_1$ $(\boldsymbol{u}_1, \boldsymbol{u}_3, t_{2,f}, t_{3,f})$. It simplifies the optimal control problem, improves the convergence performance of the algorithm, and expands the feasible domain.

The discrete form of (22) and (23) can be expressed as

$$\boldsymbol{v}_p[n] = \boldsymbol{v}_p[0] + \frac{\left(t_{p,f} - t_{p,0}\right)}{2}\left[\boldsymbol{R}_{[n+1,\cdot]} \cdot \boldsymbol{Y} \cdot \boldsymbol{G}\left(\boldsymbol{X}_p^k, \boldsymbol{U}_p\right)\right]^{\top} \tag{24}$$

$$\begin{aligned}\boldsymbol{r}_p[n] = \boldsymbol{r}_p[0] &+ \frac{\left(t_{p,f} - t_{p,0}\right)}{2}\left(\tau_p[n] + 1\right)\boldsymbol{v}_p[0] \\ &+ \frac{\left(t_{p,f} - t_{p,0}\right)^2}{4}\left[\boldsymbol{R}_{[n+1,\cdot]} \cdot \boldsymbol{Y} \cdot \boldsymbol{R} \cdot \boldsymbol{Y} \cdot \boldsymbol{G}\left(\boldsymbol{X}_p^k, \boldsymbol{U}_p\right)\right]^{\top}\end{aligned} \tag{25}$$

where the discrete nodes $n = 1, 2, \ldots, N$, the coefficient matrices $\boldsymbol{R}$ and $\boldsymbol{Y}$ are constants when the number of discrete nodes is fixed, and their computation method can be seen in [41]. $\boldsymbol{R}_{[n+1,\cdot]}$ denotes the $(n+1)$th row of $\boldsymbol{R}$.

The force function matrix is given by

$$\boldsymbol{G}\left(\boldsymbol{X}_p^k, \boldsymbol{U}_p\right) = \left[\boldsymbol{f}_{v,p}\left(\boldsymbol{x}_p^k[0], \boldsymbol{u}_p[0]\right), \ldots, \boldsymbol{f}_{v,p}\left(\boldsymbol{x}_p^k[N], \boldsymbol{u}_p[N]\right)\right]^{\top}. \tag{26}$$

Since the terminal constraints are functions with respect to the state quantities, we need to convert them into functions of the control quantities and time after using the MCPI method. To solve the problem in the SCP framework, we linearize the nonconvex terminal constraints $\boldsymbol{\phi}(\boldsymbol{x}_{3,f})$ with the first-order Taylor expansion, which yields

$$\begin{aligned}&\boldsymbol{\phi}'(\boldsymbol{x}_3[N]) \\ &\approx \boldsymbol{\phi}\left(\boldsymbol{x}_3^k[N]\right) + \nabla_{\boldsymbol{x}}\boldsymbol{\phi}\left(\boldsymbol{x}_3^k[N]\right) \\ &\quad \times \left[\frac{\partial \boldsymbol{x}_3[N]}{\partial \boldsymbol{u}_1}\left(\boldsymbol{u}_1 - \boldsymbol{u}_1^k\right)\right. \\ &\qquad \left. + \frac{\partial \boldsymbol{x}_3[N]}{\partial \boldsymbol{u}_3}\left(\boldsymbol{u}_3 - \boldsymbol{u}_3^k\right) + \sum_{p=2}^{3} \frac{\partial \boldsymbol{x}_3[N]}{\partial t_{p,f}}\left(t_{p,f} - t_{p,f}^k\right)\right] = 0\end{aligned} \tag{27}$$

where

$$\nabla_{\boldsymbol{x}}\boldsymbol{\phi}\left(\boldsymbol{x}_3^k[N]\right) = \left.\frac{\partial \boldsymbol{\phi}(\boldsymbol{x}_3[N])}{\partial \boldsymbol{x}_3[N]}\right|_{\boldsymbol{x}_3[N] = \boldsymbol{x}_3^k[N]}. \tag{28}$$

However, it is difficult to satisfy this terminal equality constraint (8)–(12), and it can encounter artificially infeasible situations. To address this issue, a 5-D slack variable $\boldsymbol{\Delta}_{\boldsymbol{\phi}}$ is introduced to relax the terminal constraints

$$|\boldsymbol{\phi}'(\boldsymbol{x}_3[N])| \leq \boldsymbol{\Delta}_{\boldsymbol{\phi}}. \tag{29}$$

To limit the size of $\boldsymbol{\Delta}_{\boldsymbol{\phi}}$, the objective function of the original trajectory optimization problem becomes

$$J_5 = J_1 + J_{\boldsymbol{\phi}} \tag{30}$$

where $J_{\boldsymbol{\phi}} = \boldsymbol{\omega}_{\boldsymbol{\phi}}^{\top}\boldsymbol{\Delta}_{\boldsymbol{\phi}}$ is the part related to the terminal constraint penalty term in the objective function, and $\boldsymbol{\omega}_{\boldsymbol{\phi}}$ is a positive penalty parameter.

The application of slack variables not only relaxes the original strict equality constraints but also reveals that the objective function $J_{\boldsymbol{\phi}}$ here is structurally similar to the objective function in P3. Therefore, the SJTR method discovers and uses this characteristic, adjusting penalty coefficients to preferentially optimize different orbital parameters.

Specifically, according to the target orbit redecision principle, we should set the penalty coefficients in $\boldsymbol{\omega}_{\boldsymbol{\phi}}$ corresponding to the semi-major axis and eccentricity to larger values. This ensures that the requirement for the semi-major axis of the circular orbit is prioritized when solving the problem. Usually, $\boldsymbol{\omega}_{\boldsymbol{\phi}}$ is an empirical parameter. As long as it is selected based on the above principle, it is easy to find a parameter suitable for this problem. This way, we can achieve the joint optimization of target orbit that satisfies the redecision principle and flight trajectory directly, without the need to trigger step-by-step optimization. Since the original constraints have been relaxed and the weight of J_1 in the objective function is small, this essentially represents a suboptimal method. Such suboptimal methods are permissible in this article, as it has been mentioned earlier that ensuring convergence and efficiency are more important than optimality.

Let $\boldsymbol{\Lambda} = [\boldsymbol{u}_1{}^\top, \boldsymbol{u}_3{}^\top, t_{2,f}, t_{3,f}]^\top$. Trust-region constraints are imposed to ensure the feasibility of linearization

$$|\boldsymbol{\Lambda}_1 - \boldsymbol{\Lambda}_1^k| \leq \boldsymbol{\delta} \tag{31}$$

where $\boldsymbol{\Lambda}_1^k$ is the solution of the previous optimization iteration and $\boldsymbol{\delta}$ defines the radius of trust-region. To further improve the convergence performance, we adopt the adaptive trust-region strategy mentioned in [42], which adjusts the size of the trust-region radius at different iteration stages.

The nonconvex constraints have been transformed into a convex optimization problem that can be solved within the SCP framework by means of lossless convexification and successive linearization. In summary, the optimal control problem constructed using the SJTR method is as follows.

P5:

Minimize: Equation (30).

Subject to: Equations (14), (29), and (31).

Using the SJTR method may lead to the following scenarios and corresponding types of orbits.

1) *Original Target Orbit:* The slack variable $\boldsymbol{\Delta}_\phi$ converges to a value that can be considered equal to zero, meaning the actual terminal conditions deviate negligibly from the target orbit.
2) *Rescue Orbit Type I:* Due to the large penalty coefficients associated with the semi-major axis and eccentricity errors, the corresponding slack variables can converge to negligible values. However, the errors in the orbital inclination and longitude of ascending node are nonnegligible and are considered primary performance indicators in the optimization problem.
3) *Rescue Orbit Type II:* All the slack variables ultimately converge to nonnegligible values, but the minimum safe altitude requirement can be satisfied, and the optimization is still dominated by adjusting the altitude.
4) *Mission Failure:* The obtained slack variables are so large that the maximum altitude is less than a_{safe}, or the optimization results are infeasible.

C. Learning-Based Warm-Start

An accurate guess of the terminal time is crucial for optimization. Since the fault occurrence time and fault conditions are generally unknown, it is also difficult to estimate the moment when the propellant is depleted. This will affect the solution difficulty of the optimization problem. Although SJTR has exhibited good convergence performance, an accurate initial guess can effectively improve the reliability and efficiency of the solution and prevent infeasible situations in extreme cases. Therefore, we designed the learning-based warm-start method for SJTR to avoid using random initial time guesses. The overall scheme is shown in Fig. 2.

This learning-based warm-start method can be divided into offline and online components. In the offline part, the dataset is split into training data and validation data, which are used to train the neural network model and evaluate the effectiveness

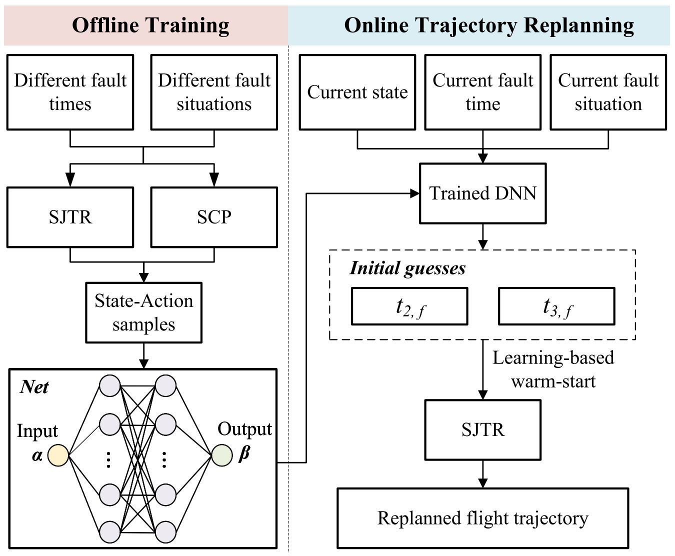


Fig. 2. Learning-based warm-start SJTR method.

of the trained model, respectively. To ensure the generalization ability of the neural network, the solutions obtained using the SJTR and the SCP methods to solve optimization problems under various fault scenarios are used as the dataset to train the neural network, and only the results of successful optimization are regarded as valid data. Thus, a mapping relationship is established among the current state, the fault scenarios, and the final times of the coasting and orbiting phases. In the online part, the pretrained mapping allows for rapid determination of flight time for these two phases, providing a more accurate initial guess for subsequent trajectory replanning. The DNN's input $\boldsymbol{\alpha}$ and output $\boldsymbol{\beta}$ are as follows:

$$\boldsymbol{\alpha} = \left[\boldsymbol{x}_{\text{fail}}^\top, t_{\text{fail}}, \eta^*\right]^\top \tag{32}$$

$$\boldsymbol{\beta} = \left[t_{2,f}, t_{3,f}\right]^\top \tag{33}$$

where $\boldsymbol{x}_{\text{fail}}$ is the state vector of the MAV at the time of fault.

To train effectively and achieve rapid convergence, the Z-score normalization method is used to standardize both the input and output [43]. The process is as follows:

$$z = \frac{\zeta - \bar{\zeta}}{\sigma} \tag{34}$$

where $\bar{\zeta}$ and σ denote the mean and standard deviation of the training dataset ζ, respectively.

The designed DNN is a fully connected feedforward neural network with an input layer, multiple hidden layers, and an output layer. The number of hidden layers and the number of neurons in each hidden layer are hyperparameters that need to be optimized. Too many hidden layers or neurons will increase the training time and may lead to overfitting, while too few hidden layers or neurons will result in underfitting. After verification and analysis, the neural network in this article is structured with three hidden layers. The first and second hidden layers each have 128 neurons, while the third hidden layer has 64 neurons. The output layer has two neurons for the final prediction.

TABLE II
MAV PARAMETERS

Parameters	Value	Parameters	Value
$m_{1,\text{dry}}$, kg	27.6	$m_{2,\text{dry}}$, kg	70.4
$m_{1,\text{prop}}$, kg	196	$m_{2,\text{prop}}$, kg	51
$I_{1,\text{sp}}$, s	293	$I_{2,\text{sp}}$, s	315
T_1, N	9000	T_2, N	800
$t_{1,\text{std}}$, s	64.17	$t_{2,\text{std}}$, s	196.93

We choose the sigmoid function as the activation function and the mean absolute error (MAE) to evaluate the deviation between predicted optimization values and outputs, as shown in (35). Moreover, the adaptive momentum (Adam) algorithm is used to minimize the loss function

$$\text{MAE} = \frac{1}{n_s \times d} \sum_{i=1}^{n_s} \sum_{j=1}^{d} \left| \beta_{ij} - \beta_{ij}^* \right| \tag{35}$$

where n_s represents the number of samples, and d is the dimension of the output. β_{ij} and β_{ij}^* are the neural network predicted value and the true value for the jth dimension of the ith sample, respectively.

In this way, when a fault occurs, we can predict the flight time of the coasting and orbiting phases for the current state based on the time of the fault and the fault situation. The computational time for this process is often negligible. We do not require highly accurate predictions; as long as they are in the vicinity of the optimal value, they can provide a good warm-start for the SJTR algorithm. This ensures the convergence performance of the algorithm and improves the accuracy of the solution results.

IV. Simulations and Analysis

In this section, the simulation results of applying the learning-based warm-start SJTR method to the trajectory replanning problem of the MAV after two typical thrust drop faults, namely, the mass flow rate drop and the specific impulse drop, are presented. The simulations are implemented on a desktop with an Intel Core i5-11600KF at 3.90 GHz. The assignment of MAV parameters is displayed in Table II, where $m_{(\cdot),\text{dry}}$, $m_{(\cdot),\text{prop}}$, $I_{(\cdot),\text{sp}}$, $T_{(\cdot)}$, and $t_{(\cdot),\text{std}}$ correspond to the dry mass, the propulsion mass, the specific impulse, the thrust magnitude, and the burning time without faults, respectively. The payload mass of the MAV $m_{\text{payload}} = 5$ kg, and the total mass m_0 at takeoff is 350 kg.

The number of Chebyshev nodes is $N = 100$, and the initial radius of trust-region constraint is $\boldsymbol{\delta} = [0.5, 0.5, 6, 6]^\top$. During optimization iterations, if the results of two consecutive iterations are less than the convergence tolerance $\boldsymbol{\epsilon_x} = [1\text{m}, 1\text{m}, 1\text{m}, 0.1\text{m/s}, 0.1\text{m/s}, 0.1\text{m/s}, 0.1\text{kg}]^\top$, $\boldsymbol{\epsilon_{u_1}} = \boldsymbol{\epsilon_{u_3}} = [0.001, 0.001, 0.001]^\top$, and $\epsilon_{t_{2,f}} = \epsilon_{t_{3,f}} = 0.01$s, the optimization process is considered to have successfully concluded. The penalty coefficients for terminal constraints are set to $\boldsymbol{\omega_\phi} = [10^7, 10^7, 10^7, 6 \times 10^5, 6 \times 10^5]^\top$. The selection of this parameter is based on the target orbit redecision principles and then determined empirically. Therefore, the penalty coefficients corresponding to the semi-major axis and eccentricity are set to relatively large values.

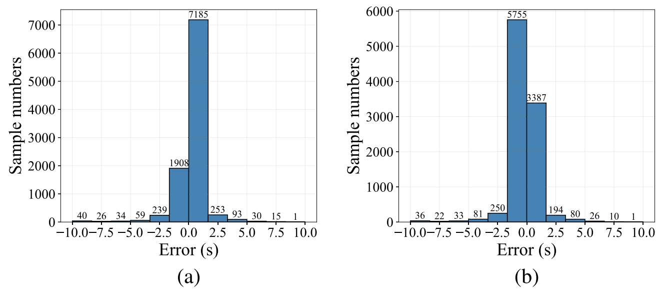


Fig. 3. Error histograms of the DNN in the test set. Errors of (a) $t_{2,f}$ and (b) $t_{3,f}$.

The initial position of the MAV in the MCI coordinate system is $\boldsymbol{r}_{1,0} = [-303.103, -3374.249, 238.074]^\top$ km. We set the parameters of the target orbit as $a^* = (300 + R_M)$ km, $e^* = 0$, $i^* = 29.5°$, and $\Omega^* = 253.2°$, where $R_M = 3396.19$ km represents the radius of Mars, and the minimum safe orbital altitude is $a_{\text{safe}} = 250$ km.

A. Mass Flow Rate Drop Faults ($L = 0$)

1) Analysis of the Results for Learning-Based Warm-Start SJTR Method: To initiate the algorithm, we set the initial trajectory guess as a linear interpolation from the starting point to the ending point. We conducted 100000 sets of Monte Carlo simulations using the SJTR method (70%) and the SCP method (30%) to generate an offline DNN dataset. In each set, the data come from random fault occurrence time $t_{\text{fail}} \in [0, t_{1,\text{std}}]$ and remaining thrust percentage $\eta^* \in [0.2, 0.55]$, both of which follow a uniform distribution. The data are divided into a training set consisting of 90 000 trajectories and a validation set consisting of 10 000 trajectories. There are a total of 50 batches in training, and the number of samples per batch is set to 128. The learning rate is set as 0.01.

The estimation error distributions of $t_{2,f}$ and $t_{3,f}$ in the validation set are shown in Fig. 3. It can be seen that the estimation error is mainly concentrated and distributed in a range around 0 s, so this DNN can establish a good mapping relationship from the fault state to the time of coasting and orbiting phases.

To verify the effectiveness of the learning-based warm-start SJTR method, we conducted 5000 sets of Monte Carlo simulations with the same fault conditions distribution as previously mentioned. When the MAV detects a fault, the initial time guesses are obtained using the trained network. Subsequently, the SJTR is executed and the corresponding flight trajectory and orbit type are optimized at the same time. The results of the solution using the SJTR method and the learning-based warm-start SJTR method are shown in Fig. 4.

In this result, each point represents a fault condition characterized by the combination of fault occurrence time and remaining thrust percentage. The four colors A, B, C, and D correspond to four orbit types: original target orbit, rescue orbit type I, rescue orbit type II, and mission failure, respectively.

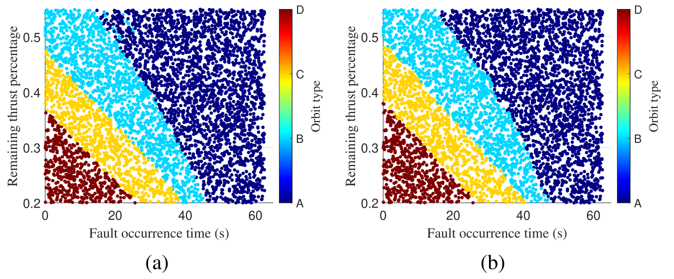


Fig. 4. Orbit type statistical graph (a) SJTR and (b) learning-based warm-start SJTR.

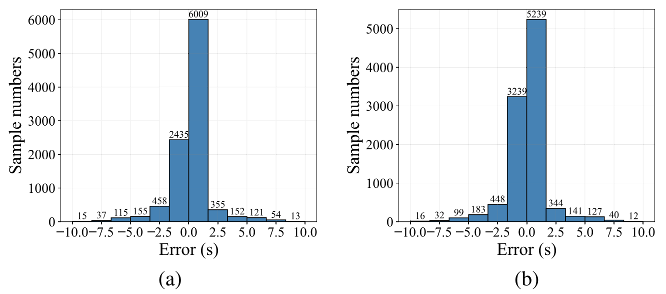


Fig. 5. Error histograms of the DNN in the test set. Errors of (a) $t_{2,f}$ and (b) $t_{3,f}$.

TABLE III
CASES FOR FOUR ORBIT TYPES

	Case 1	Case 2	Case 3	Case 4	Unit
Orbit type	A	B	C	D	-
$r_{x,\text{fail}}$	-288.64	-283.16	-300.17	-295.04	km
$r_{y,\text{fail}}$	-3380.76	-3384.03	-3374.99	-3377.26	km
$r_{z,\text{fail}}$	245.32	249.15	238.75	241.28	km
$v_{x,\text{fail}}$	748.27	886.07	383.60	565.85	m/s
$v_{y,\text{fail}}$	-432.83	-545.33	-137.37	-284.58	m/s
$v_{z,\text{fail}}$	504.53	638.88	144.53	325.30	m/s
m_{fail}	255.84	234.81	320.38	286.46	kg
t_{fail}	30.06	36.78	9.46	20.29	s
η^*	46.84	24.12	34.34	20.81	%

The average runtime of the learning-based warm-start SJTR is 0.447 s, and the average number of iterations is 24.38. Compared with the SJTR, the efficiency is improved by 5.2%, and the number of iterations is reduced by 1.2. The improvement in computational performance is limited because the SJTR already has an excellent convergence performance. As shown in Fig. 4(a), the SJTR without using the warm-start approach can result in some unsmooth boundaries and inaccurate orbit-type determinations. This is because, in some fault situations corresponding to ambiguous orbit types, poor initial guesses prevent the SJTR from finding a better solution or lead to convergence difficulties. Thus, the learning-based warm-start improves the performance of the algorithm by providing a better initial guess for SJTR.

2) Comparison of Trajectory Replanning Methods: To demonstrate the effectiveness of the learning-based warm-start SJTR, we compare it with the mainstream downgrading [20] and upgrading [19] methods for solving the fault replanning problem through four different fault cases shown in Table III. According to Fig. 1, the downgrading method can be divided into Steps 1–3, which, respectively, correspond to solving P4, P3, and P2. The upgrading method can be divided into Steps 1–2, which, respectively,correspond to solving P2 and P3. The comparison results of these methods are shown in Table IV.

For Case 1, it corresponds to entering the original target orbit. All the methods show that the deviations in altitude Δh_f, orbital inclination Δi_f, and longitude of ascending node $\Delta\Omega_f$ from the original target orbit are all zero. Based on the optimized final mass of the MAV, it can be seen that compared with the downgrading method, the proposed method sacrifices some optimality. This characteristic facilitates the joint trajectory replanning method designed in this article. The reason why the upgrading method results in a lower final mass compared with the other two methods is that the term related to fuel optimality is not introduced into its performance index.

For Case 2, it corresponds to entering the rescue orbit type I. The time required for the downgrading and upgrading methods from detecting the fault occurrence to planning a new trajectory is 1.555 and 1.236 s, respectively, while the time required for the proposed method is only 0.605 s, which is an improvement of approximately 50%. Moreover, the parameters of the final rescue orbit are not much different.

For Case 3, it corresponds to entering the rescue orbit type II. The final altitude from the target orbit obtained by the downgrading and upgrading methods is both −29.40 km, and the result of the proposed method is −33.36 km. All of them meet the minimum safety orbit requirements. In the downgrading method, although no feasible solution is found in Step 2, the calculation is still necessary, rendering its runtime statistics meaningless. Therefore, it takes at least 1.521 s (denoted as 1.521^+) to solve this situation. The calculation efficiency of the upgrading method is good, but it is still inferior to our proposed method.

For Case 4, the situation corresponds to a mission failure. In this case, the maximum orbital altitude that the MAV can reach is less than the safe orbital altitude. The downgrading method and the upgrading method take 1.556^+ and 0.711 s, respectively, to determine this situation, while the proposed method only takes 0.617 s.

B. Specific Impulse Drop Faults ($L = 1$)

1) Analysis of the Results for Learning-Based Warm-Start SJTR Method: We generate and train a dataset in the same way and with the same neural network architecture as in Section III within the ranges of the random fault occurrence time $t_{\text{fail}} \in [0, t_{1,\text{std}}]$ and the remaining thrust percentage $\eta^* \in [0.8, 1)$. The prediction accuracies of the trained neural network for the end times of the two flight phases are shown in Fig. 5.

The results of 5000 sets of Monte Carlo simulations using the SJTR method and the learning-based warm-start SJTR method are shown in Fig. 6. The average runtime of the learning-based warm-start SJTR is 0.454 s, and the average number of iterations is 18.69. Compared with the SJTR,

TABLE IV
COMPARISON OF TRAJECTORY REPLANNING RESULTS FOR MASS FLOW RATE DROP FAULTS

Case	Index	Downgrading Method			Upgrading Method		Ours
		Step 1	Step 2	Step 3	Step 1	Step 2	
1	$m_{3,f}$ (kg)	75.75	–	–	75.46	75.45	75.69
	Δh_f (km)	0	–	–	0	0	0
	Δi_f (deg)	0	–	–	-0.321	0	0
	$\Delta \Omega_f$ (deg)	0	–	–	3.248	0	0
	Total Runtime (s)		0.755		1.193		0.547
2	$m_{3,f}$ (kg)	74.60	75.40	–	75.42	75.40	75.40
	Δh_f (km)	0	0	–	0	0	-0.029
	Δi_f (deg)	0	-0.489	–	2.041	-0.489	-0.476
	$\Delta \Omega_f$ (deg)	0	1.685	–	4.776	1.685	1.633
	Total Runtime (s)		1.555		1.236		0.605
3	$m_{3,f}$ (kg)	72.22	Infeasible	75.40	75.40	–	75.40
	Δh_f (km)	0	Infeasible	-29.40	-29.40	–	-33.36
	Δi_f (deg)	0	Infeasible	-9.591	-9.591	–	-1.076
	$\Delta \Omega_f$ (deg)	0	Infeasible	0.631	0.631	–	4.175
	Total Runtime (s)		1.521^{+}		0.779		0.621
4	$m_{3,f}$ (kg)	68.60	Infeasible	75.40	75.40	–	75.40
	Δh_f (km)	0	Infeasible	-70.81	-70.81	–	-78.78
	Δi_f (deg)	0	Infeasible	-2.099	-2.099	–	-0.942
	$\Delta \Omega_f$ (deg)	0	Infeasible	3.916	3.916	–	3.809
	Total Runtime (s)		1.556^{+}		0.711		0.617

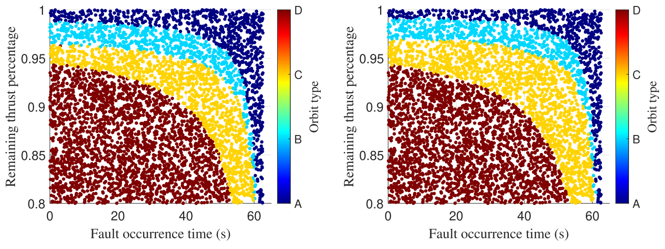


Fig. 6. Orbit type statistical graph (a) SJTR and (b) learning-based warm-start SJTR.

TABLE V
CASES FOR FOUR ORBIT TYPES

	Case 5	Case 6	Case 7	Case 8	Unit
Orbit type	A	B	C	D	-
$r_{x,\text{fail}}$	-257.02	-294.69	-281.73	-301.17	km
$r_{y,\text{fail}}$	-3401.44	-3377.45	-3384.94	-3374.66	km
$r_{z,\text{fail}}$	269.67	241.50	250.25	238.41	km
$v_{x,\text{fail}}$	1470.65	576.04	919.16	339.41	m/s
$v_{y,\text{fail}}$	-1027.30	-293.85	-573.94	-102.19	m/s
$v_{z,\text{fail}}$	1206.95	337.57	675.10	101.14	m/s
m_{fail}	163.99	284.46	229.74	329.08	kg
t_{fail}	59.39	20.92	38.40	6.68	s
η^*	92.99	97.09	92.62	92.41	%

the efficiency is improved by 11.3%, and the number of iterations is reduced by 3.4. In comparison with the mass flow rate drop fault shown in Fig. 4, when a specific impulse drop fault represented by fuel leakage occurs, the proportion of allowable thrust loss is very small. This is because the flight time will be correspondingly extended when $L = 0$, which can be understood as the fuel not being wasted. However, not all the consumed fuel is used to provide thrust when $L = 1$.

The proposed SJTR method can effectively handle the replanning problem after such a fault. However, due to inaccurate initial guesses, the solution may occasionally fail. Therefore, the learning-based warm-start SJTR method effectively overcomes this drawback and makes the replanning results more reliable.

2) Comparison of Trajectory Replanning Methods: We set up four scenarios for the specific impulse drop fault, namely, Cases 5–8 as shown in Table V, to compare the differences in solution results and efficiency between the learning-based warm-start SJTR method and other fault replanning methods.

As can be seen from Table VI, the results obtained by the proposed method are not optimal solutions. Moreover, when the MAV cannot enter the original target orbit after a fault occurs, the altitude error is larger than that of the other two methods, while the orbital inclination error is smaller. This is related to the penalty coefficients of the terminal constraints we set, which are chosen empirically. To correctly determine the type of orbit to enter after a fault, we have found that as long as we follow the target orbit redecision principles, the selection of penalty coefficients is not difficult, even though different penalty coefficients may affect the final orbit entry accuracy.

It is difficult to directly assess the superiority of the downgrading method and the upgrading method from the simulation results. When various fault scenarios occur, the decision-making sequences of these methods have a significant impact on the replanning efficiency. For example, in the case of a minor fault, the downgrading method can perform planning at a relatively fast speed. However, in the event of a more

TABLE VI
COMPARISON OF TRAJECTORY REPLANNING RESULTS FOR SPECIFIC IMPULSE DROP FAULTS

Case	Index	Downgrading Method			Upgrading Method		Ours
		Step 1	Step 2	Step 3	Step 1	Step 2	
5	$m_{3,f}$ (kg)	75.80	–	–	75.48	75.48	75.79
	Δh_f (km)	0	–	–	0	0	0
	Δi_f (deg)	0	–	–	5.216	0	0
	$\Delta\Omega_f$ (deg)	0	–	–	5.159	0	0
	Total Runtime (s)		0.729		1.275		0.617
6	$m_{3,f}$ (kg)	74.45	75.40	–	75.43	75.40	75.40
	Δh_f (km)	0	0	–	0	0	-0.103
	Δi_f (deg)	0	-0.479	–	-2.087	-0.479	-0.403
	$\Delta\Omega_f$ (deg)	0	1.651	–	3.829	1.651	1.341
	Total Runtime (s)		1.527		1.239		0.588
7	$m_{3,f}$ (kg)	73.37	Infeasible	75.40	75.40	–	75.40
	Δh_f (km)	0	Infeasible	-18.19	-18.19	–	-23.54
	Δi_f (deg)	0	Infeasible	3.837	3.837	–	-0.751
	$\Delta\Omega_f$ (deg)	0	Infeasible	5.610	5.610	–	2.560
	Total Runtime (s)		1.366^{+}		0.769		0.606
8	$m_{3,f}$ (kg)	70.98	Infeasible	75.40	75.40	–	75.40
	Δh_f (km)	0	Infeasible	-78.55	-78.55	–	-103.85
	Δi_f (deg)	0	Infeasible	-14.03	-14.03	–	-1.257
	$\Delta\Omega_f$ (deg)	0	Infeasible	-2.978	-2.978	–	4.347
	Total Runtime (s)		1.758^{+}		1.023		0.644

severe fault, multiple steps of judgment are necessitated, and there is a possibility of encountering infeasible solutions during this process. The upgrading method behaves in the opposite way in this situation. If the longest time taken by each method to solve all the cases is regarded as the calculation time for that method, the proposed method is at least 49.5% more efficient than the other two trajectory replanning schemes.

In addition to its computational efficiency advantage, the most significant aspect of the proposed method is that it enables replanning without the need for complex step-by-step judgments. Moreover, the general trajectory replanning method may encounter the issue of infeasible solution propagation, while the proposed method rarely encounters infeasible solutions, thereby significantly enhances the system's reliability.

V. CONCLUSION

This article proposes the SJTR method for optimizing the target orbit and flight trajectory of a MAV after encountering a mass flow rate drop fault and a specific impulse drop fault in the propulsion system during the flight. The method integrates the MCPI framework, known for its good convergence performance, and reveals the feature that the optimization problem can adhere to the orbit redecision principle by designing penalty coefficients for terminal constraints. In addition, by establishing a DNN that maps fault situations to time optimization variables, the SJTR method is equipped with the learning-based warm-start strategy that overcomes the problem of inaccurate orbit type determination. The simulation results show that compared with other trajectory replanning methods after faults, the proposed method eliminates the need for step-by-step decision-making, offering higher computational efficiency and solution feasibility. It provides a nonoptimal but highly reliable solution for hazardous Mars ascent missions. In future work, we will further consider the impact of DNN structures on the learning scheme proposed in this article, including the design of basic architectures and other more advanced neural network models.

REFERENCES

[1] Y. Zhang, Y. Guo, G. Ma, and B. Wie, "Fixed-time pinpoint Mars landing using two sliding-surface autonomous guidance," *Acta Astronautica*, vol. 159, pp. 547–563, Jun. 2019.

[2] I.-S. Chang, "Space launch vehicle reliability," *Crosslink*, vol. 2, no. 1, pp. 22–32, Mar. 2001.

[3] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[4] F. Gettatelli, B. Benedikter, A. Zavoli, S. Pizzurro, and E. Cavallini, "Convex optimization of ascent and powered descent of a reusable launch vehicle," in *Proc. AIAA SCITECH Forum*, Jan. 2023, p. 2644.

[5] B. Benedikter et al., "A convex approach to rocket ascent trajectory optimization," in *Proc. 8th Eur. Conf. Aeronaut. Space Sci. (EUCASS)*, 2019, pp. 1–15.

[6] K. Li, Y. Guo, G. Ran, Y. Lyu, and G. Ma, "Convex optimization-based model predictive control for Mars ascent vehicle guidance system," *IEEE/CAA J. Autom. Sinica*, early access, Dec. 24, 2024, doi: 10.1109/JAS.2024.124587.

[7] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, and C. L. Philip Chen, "Review of advanced guidance and control algorithms for space/aerospace vehicles," *Prog. Aerosp. Sci.*, vol. 122, Apr. 2021, Art. no. 100696.

[8] B. Açikmese, J. M. Carson, and L. Blackmore, "Lossless convexification of nonconvex control bound and pointing constraints of the soft landing optimal control problem," *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 6, pp. 2104–2113, Nov. 2013.

[9] M. Szmuk, T. P. Reynolds, and B. Açıkmeşse, "Successive convexification for real-time six-degree-of-freedom powered descent guidance with state-triggered constraints," *J. Guid., Control, Dyn.*, vol. 43, no. 8, pp. 1399–1413, Aug. 2020.

[10] H. Hong, A. Maity, F. Holzapfel, and S. Tang, "Model predictive convex programming for constrained vehicle guidance," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 55, no. 5, pp. 2487–2500, Oct. 2019.

[11] Y. Ma, B. Pan, and R. Yan, "Picard iteration-based convexification for fuel-optimal rocket descent inside atmosphere," *J. Guid., Control, Dyn.*, vol. 46, no. 2, pp. 343–349, Feb. 2023.

[12] Y. Ma, B. Pan, C. Hao, and S. Tang, "Improved sequential convex programming using modified Chebyshev–Picard iteration for ascent trajectory optimization," *Aerosp. Sci. Technol.*, vol. 120, Jan. 2022, Art. no. 107234.

[13] B. Benedikter, A. Zavoli, G. Colasurdo, S. Pizzurro, and E. Cavallini, "Convex approach to three-dimensional launch vehicle ascent trajectory optimization," *J. Guid., Control, Dyn.*, vol. 44, no. 6, pp. 1116–1131, Jun. 2021.

[14] H. Chen, H. Luo, B. Huang, B. Jiang, and O. Kaynak, "Transfer learning-motivated intelligent fault diagnosis designs: A survey, insights, and perspectives," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 3, pp. 2969–2983, Mar. 2024.

[15] G. Ran, H. Chen, C. Li, G. Ma, and B. Jiang, "A hybrid design of fault detection for nonlinear systems based on dynamic optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 9, pp. 5244–5254, Sep. 2023.

[16] H. Chen and B. Huang, "Explainable fault diagnosis using invertible neural networks-part I: A left manifold-based solution," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 7, pp. 12758–12771, Jul. 2025.

[17] Z. Song, Y. Liu, Y. He, and C. Wang, "Autonomous mission reconstruction during the ascending flight of launch vehicles under typical propulsion system failures," *Chin. J. Aeronaut.*, vol. 35, no. 6, pp. 211–225, Jun. 2022.

[18] Y. Ma, B. Pan, and S. Tang, "Improved parallel-structured Newton-type guidance for launch vehicles under thrust drop fault," *J. Spacecraft Rockets*, vol. 59, no. 2, pp. 467–481, Mar. 2022.

[19] Z. Song, C. Wang, and Q. Gong, "Joint dynamic optimization of the target orbit and flight trajectory of a launch vehicle based on state-triggered indices," *Acta Astronautica*, vol. 174, pp. 82–93, Sep. 2020.

[20] X. Miao, L. Cheng, Z. Zhang, J. Li, and S. Gong, "Convex optimization for post-fault ascent trajectory replanning using auxiliary phases," *Aerosp. Sci. Technol.*, vol. 138, Jul. 2023, Art. no. 108336.

[21] C. Sánchez-Sánchez and D. Izzo, "Real-time optimal control via deep neural networks: Study on landing problems," *J. Guid., Control, Dyn.*, vol. 41, no. 5, pp. 1122–1135, May 2018.

[22] Z. Sun, J. Simo, and S. Gong, "Satellite attitude identification and prediction based on neural network compensation," *Space: Sci. Technol.*, vol. 3, p. 0009, Jan. 2023.

[23] R. Chai, A. Tsourdos, A. Savvaris, Y. Xia, and S. Chai, "Real-time reentry trajectory planning of hypersonic vehicles: A two-step strategy incorporating fuzzy multiobjective transcription and deep neural network," *IEEE Trans. Ind. Electron.*, vol. 67, no. 8, pp. 6904–6915, Aug. 2020.

[24] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, and C. L. P. Chen, "Six-DOF spacecraft optimal trajectory planning and real-time attitude control: A deep neural network-based approach," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 5005–5013, Nov. 2020.

[25] H. Hua and Y. Fang, "A novel learning-based trajectory generation strategy for a quadrotor," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 7, pp. 9068–9079, Jul. 2024.

[26] X. He, S. Tan, Z. Wu, and L. Zhang, "Mission reconstruction for launch vehicles under thrust drop faults based on deep neural networks with asymmetric loss functions," *Aerosp. Sci. Technol.*, vol. 121, Feb. 2022, Art. no. 107375.

[27] H. Li, Z. Wang, C. Lan, P. Wu, and N. Zeng, "A novel dynamic multiobjective optimization algorithm with non-inductive transfer learning based on multi-strategy adaptive selection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 11, pp. 16533–16547, Nov. 2024.

[28] H. Li, Z. Wang, N. Zeng, P. Wu, and Y. Li, "Promoting objective knowledge transfer: A cascaded fuzzy system for solving dynamic multiobjective optimization problems," *IEEE Trans. Fuzzy Syst.*, vol. 32, no. 11, pp. 6199–6213, Nov. 2024.

[29] H. Wu, Y. Qiao, and X. Luo, "A fine-grained regularization scheme for non-negative latent factorization of high-dimensional and incomplete tensors," *IEEE Trans. Services Comput.*, vol. 17, no. 6, pp. 3006–3021, Nov. 2024.

[30] S. Banerjee et al., "Learning-based warm-starting for fast sequential convex programming and trajectory optimization," in *Proc. IEEE Aerosp. Conf.*, Mar. 2020, pp. 1–8.

[31] B. Xu, Y. Shou, Z. Shi, and T. Yan, "Predefined-time hierarchical coordinated neural control for hypersonic reentry vehicle," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, pp. 8456–8466, Nov. 2023.

[32] Z. Li, Y. Xia, C.-Y. Su, J. Deng, J. Fu, and W. He, "Missile guidance law based on robust model predictive control using neural-network optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 8, pp. 1803–1809, Aug. 2015.

[33] Z. Song, Q. Gong, C. Wang, Y. He, and G. Shi, "Review and progress of the autonomous guidance method for long March launch vehicle ascent flight," *SCIENTIA SINICA Informationis*, vol. 51, no. 10, pp. 1587–1608, Oct. 2021.

[34] D. Chao, R. Qi, and B. Jiang, "Adaptive fault-tolerant control for the ascent phase of hypersonic vehicle with time-varying full state constraints," *Aerosp. Sci. Technol.*, vol. 131, Dec. 2022, Art. no. 108006.

[35] R. Chai, Y. Guo, Z. Zuo, K. Chen, H.-S. Shin, and A. Tsourdos, "Cooperative motion planning and control for aerial-ground autonomous systems: Methods and applications," *Prog. Aerosp. Sci.*, vol. 146, Apr. 2024, Art. no. 101005.

[36] P. Lu, H. Sun, and B. Tsai, "Closed-loop endoatmospheric ascent guidance," *J. Guid., Control, Dyn.*, vol. 26, no. 2, pp. 283–294, Mar. 2003.

[37] X. Liu, P. Lu, and B. Pan, "Survey of convex optimization for aerospace applications," *Astrodynamics*, vol. 1, no. 1, pp. 23–40, Sep. 2017.

[38] X. Miao, Y. Song, Z. Zhang, and S. Gong, "Successive convexification for ascent trajectory replanning of a multistage launch vehicle experiencing nonfatal dynamic faults," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 58, no. 3, pp. 2039–2052, Jun. 2022.

[39] X. Bai, "Modified Chebyshev-picard iteration methods for solution of initial value and boundary value problems," Ph.D. dissertation, Texas A&M Univ., College Station, TX, USA, 2010.

[40] Y. Ma, B. Pan, and R. Yan, "Feasible sequential convex programming with inexact restoration for multistage ascent trajectory optimization," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 59, no. 2, pp. 1217–1230, Apr. 2023.

[41] J. L. Junkins, A. Bani Younes, R. M. Woollands, and X. Bai, "Picard iteration, Chebyshev polynomials and Chebyshev-picard methods: Application in astrodynamics," *J. Astron. Sci.*, vol. 60, nos. 3–4, pp. 623–653, Dec. 2013.

[42] K. Li, Y. Guo, G. Ran, and J. H. Park, "Adaptive sequential convex programming for Mars ascent vehicle multiphase trajectory optimization," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 60, no. 6, pp. 9369–9382, Dec. 2024.

[43] Y. Shi and Z. Wang, "A deep learning-based approach to real-time trajectory optimization for hypersonic vehicles," in *Proc. AIAA Scitech Forum*, Jan. 2020, p. 0023.

Kun Li received the B.S. degree in automation from the College of Control Science and Engineering, China University of Petroleum, Qingdao, China, in 2020, and the M.S. degree in control science and engineering from Harbin Institute of Technology, Harbin, China, in 2022, where he is currently pursuing the Ph.D. degree with the Department of Control Science and Engineering.

His current research interests include trajectory optimization and learning-based trajectory planning.

Guangtao Ran (Member, IEEE) received the Ph.D. degree in control science and engineering from Harbin Institute of Technology, Harbin, China, in 2023.

From 2021 to 2022, he was a joint Ph.D. Student with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. He is currently an Assistant Professor with the Department of Control Science and Engineering, Harbin Institute of Technology. His current research interests include multiagent systems, optimal control, fuzzy control, fault detection, networked control systems, and security control.

Dr. Ran was a Guest Editor of *Machines* and *Actuators*.

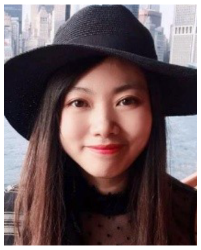
Yanning Guo received the M.S. and Ph.D. degrees in control science and engineering from Harbin Institute of Technology, Harbin, China, in 2008 and 2012, respectively.

He is currently a Professor with the Department of Control Science and Engineering, Harbin Institute of Technology and teaches and performs research in the fields of deep space exploration, satellite attitude control, and nonlinear control.

Ju H. Park (Senior Member, IEEE) received the Ph.D. degree in electronics and electrical engineering from Pohang University of Science and Technology (POSTECH), Pohang, Republic of Korea, in 1997.

From 1997 to 2000, he was a Research Associate with the Engineering Research Center-Automation Research Center, POSTECH. He joined Yeungnam University, Gyeongsan, Republic of Korea, in 2000, where he is currently the Chuma Chair Professor. He has authored and co-authored a number of articles in his research interests, which include robust control and filtering, neural/complex networks, fuzzy systems, multiagent systems, and chaotic systems.

Dr. Park is a fellow of Korean Academy of Science and Technology. He serves as an Editor for *International Journal of Control, Automation and Systems*. He is also a Subject Editor/Advisory Editor/Associate Editor/Editorial Board Member of several international journals, including *IET Control Theory and Applications*, *Applied Mathematics and Computation*, *Journal of the Franklin Institute*, *Nonlinear Dynamics*, *Engineering Reports*, *Cogent Engineering*, IEEE TRANSACTIONS ON FUZZY SYSTEMS, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and IEEE TRANSACTIONS ON CYBERNETICS.



Yao Zhang (Senior Member, IEEE) is currently with the Department of Mechanical Engineering, University College London, London, U.K. She is the Principal Investigator of several projects (£3 million in total) funded by EPSRC, Innovate U.K., and Royal Society. Her research interests cover learning-based control, offshore renewable energy systems, and AI-powered monitoring.

Dr. Zhang serves as the Deputy Editor for *Ocean Engineering* and an Associate Editor for IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, and *IET Generation, Transmission, and Distribution*. She serves as a Committee Member and local-event Coordinator for the Control System Chapter in IEEE U.K. and Ireland Section, and the Secretary of IEEE U.K. and Ireland Women in Engineering.